Analytical Techniques for the Interpretation of Satellite Derived Marine Animal Locations

Sascha Frydman

BSc (Hons), James Cook University

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

University of Tasmania

November, 2011

Declaration of Originality

This thesis contains no material which has been accepted for a degree or diploma by the University or any other institution, except by way of background information and duly acknowledged in the thesis, and to the best of the my knowledge and belief no material previously published or written by another person except where due acknowledgement is made in the text of the thesis, nor does the thesis contain any material that infringes copyright.

Signed: (Sascha Frydman)

Date:

Statement of Authority of Access

This thesis may be made available for loan and limited copying in accordance with the Copyright Act 1968.

Signed: (Sascha Frydman)

Date:

Thesis Abstract

Over the past few decades there has been a dramatic increase in the scale and complexity of data collected from free ranging animals. Advances in data collection have outstripped the development of sufficiently powerful analytical tools. As such, researchers require the ability to store, analyse, interpret and visualise large quantities of animal derived telemetry data. This is achieved through a combination of statistical analysis algorithms and accompanying data analysis software. During the austral summer of 2003/04, a multi-species animal tracking project was conducted at Heard Island. Over a period of 40 days, the off-shore movements of Antarctic fur seals, Macaroni penguins and King penguins were tracked using animal-borne transmitters and the Argos satellite system. This project required real-time analysis capabilities to integrate data from multiple predator movements with other contemporaneous data sources. This thesis describes the development of these tools, from the initial package used to direct field efforts to the creation of sophisticated mathematical approaches to the analysis and interpretation of these data. Though developed in response to the Heard Island project, the work presented here is generally applicable to all Argos derived animal movement data. The outcomes of this work are:

1) A purpose built software system that archived, analysed and visualised satellite derived location data. This system provided researchers with near real-time estimates of areas of intensive foraging activity. It was designed to support the quantitative requirements of the predator-prey study at Heard Island.

2) A concise set of equations for the generation of weighted kernel smoothed utilisation distributions (UD). These equations can directly take geographic coordinates (latitude and longitude) without the need for preliminary conversion to a Cartesian based system such as universal transverse Mercator. UDs of animal location tracks were created using weighted and non-weighted kernels and the effect of using different methods for determining smoothing parameter were explored. This study found that the predictive ability of the UD was most effective when using a weighted kernel in conjunction with likelihood cross-validation.

3) A new type of speed filter that uses an evolutionary optimisation algorithm to incorporate an elliptical location error into the positional estimates provided by Argos. When tested against the output of the commonly used filter defined in McConnell *et al*. 1992, the optimising filter was found to remove half as many locations.

4) A time weighted probability density function (PDF) of speeds travelled by multiple individuals was used to create a population level estimate of movement behaviour. This PDF was incorporated into an algorithm for interpolating movement between locations. The optimisation framework was used to guide the interpolated system so that its PDF matched that of the population level PDF. This system utilised knowledge of location error in conjunction with the population distribution. The interpolated data was then used to generate a weighted kernel density estimate or UD of the animal's movements. This method is called model interpolated kernel smoothing (MIKS). A series of validation tests were conducted to assess the performance of this MIKS to simpler forms of UD such as that created with linear interpolation. MIKS was found to out perform the other types of UD.

5) A flexible software platform for the development of location filtering and location interpolation algorithms that incorporates a form of evolutionary optimisation known as extremal optimisation. This system, written in C++, provides all the functionality required to implement the algorithms described in this thesis. It includes a database for the archival and rapid retrieval of Argos satellite data. It also provides a framework for building a spatial location analysis system. Finally, it provides links to various data analysis and presentation packages including ParaView and R.

This series of studies provides a set of analytical tools and supporting software for the interpretation of animal movement data. It utilises a novel approach to exploring probability distributions which will create new opportunities for the development of analytical techniques in the future.

Acknowledgements

The work presented in this thesis began with my involvement in the preparations for the Heard Island expedition at the end of 2003. This was my first trip 'South' and only served to further fuel what had already been a life long ambition to be involved in Antarctic research. For the numerous opportunities, faith in my abilities, inspiration, support and friendship, my utmost thanks and appreciation go to Nick Gales.

To everyone at the Antarctic Division, my friends, workmates, shipmates, thankyou for making it a great place to work for so many years. In particular, to everyone involved in the Heard Island expedition. Far too many names to mention, but the data that made this thesis possible was the result of the efforts of many dedicated, hard working people.

My supervisor at the university, Mark Hindell, gave me the opportunity to develop my ideas into an academic thesis. His critique of my writing and suggestions for improvements were invaluable.

To my friends, Zane and Leanne. Your generosity knows no limits. The assistance you have both given me in the last month of writing has made all the difference. Not only did you give me a place to stay, but you fed me and made me feel like one of the family. Thanks again.

To my wife, Andrea. Nothing I can say will ever truly convey my love and appreciation for the support and understanding you have given me throughout this very long process. This degree is as much yours as it is mine.

To Ella, I'm sorry that your Daddy kept going away. You know I'll make it up to you. Karate, school, piano, dancing, swimming, you name it, now I can be there for all of it!

Statement of Co-authorship

For the publication that was produced from Chapter 2:

Nick Gales (Australian Antarctic Division) was in charge of the expedition to Heard Island and in this capacity defined the requirements for the software package, HeardMap. I designed and developed the program. I wrote the paper while Dr. Gales, as my supervisor acted to review and suggest improvements to the text. I, the candidate, was responsible for 95% of this work while the second author, Nick Gales, contributed 5%.

We the undersigned agree with the above stated "proportion of work undertaken" for each of the above published (or submitted) peer-reviewed manuscripts contributing to this thesis:

Professor Mark Hindell
Candidate's Supervisor
Institute for Marine and Antarctic Studies
University of Tasmania

Signed: _____

Date: _____

Professor Craig Johnson
Head of School
Institute for Marine and Antarctic Studies
University of Tasmania

Signed: _____

Date: _____

Table of Contents

List of Figures

List of Tables

Chapter 1

General Introduction

1.1 Introduction

During the Austral summer of 2003/04, a large expedition to Heard Island and its surrounding waters took place. The aim of this expedition was to investigate the ecosystem dynamics, oceanography and predator-prey interactions of the resident species in the region. Over a period of 40 days, the off-shore movements of Antarctic fur seals, Macaroni penguins and King penguins were tracked using animal-borne transmitters and the Argos satellite system. A requirement for the storage, analysis and interpretation of this location data was identified. Consequently, a preliminary investigation to find an appropriate solution was conducted. This highlighted the lack of readily available analysis software that could be dedicated to the research needs of the expedition. The decision was then made to develop an in-house system for real-time use in the field as well as more intensive processing at the conclusion of the expedition.

The development of this system provided the impetus for the ongoing design of more sophisticated data driven analysis tools as well as the requisite software needed to implement these ideas. These algorithms were initially intended for a more in-depth analysis of the Heard Island data set. However, the opportunity to develop a more generic animal movement analysis package also became apparent. In this thesis, I describe the development of new techniques for the analysis and interpretation of satellite derived movement data as well as the software algorithms and applications that implement these methods. In addition to providing applications for the storage and handling of large volumes of spatial data, a series of new methods that address the spatial uncertainty in the locations and the irregular temporal gaps in acquisition times are presented. Furthermore, the development of a new technique for estimating an animal's spatial usage is also described.

1.2 Argos Satellite Tracking

For the past thirty years, the Argos satellite system has provided researchers with a reliable and simple solution for the acquisition of location data from far ranging and remotely located animals both on land and at-sea (Cauzac & Ortega 2000). Although more sophisticated and accurate technologies such as the new Fastloc GPS tags (Costa et al. 2010) are beginning to be deployed, the Argos system continues to provide the bulk of data for animal tracking research. In spite of this, consensus on the best way to analyse and interpret this data has not been reached. On the contrary, the ongoing development of affordable and increasingly more powerful computers is promoting even greater levels of research into various analytical techniques.

The Argos system is a network of polar-orbiting satellites that receive signals from battery powered, animal-borne platform terminal transmitters (PTT). The system relies on the measurement of Doppler shift frequency changes caused by the relative motion of the satellite to the terrestrial transmitter. When several transmissions are received over a period of several minutes, a series of calculations can be used to determine the approximate geographical position of the PTT (Argos 2008). Many factors effect the performance of this system. For instance, in the marine environment, animals such as seals and penguins spend a considerable amount of time underwater where the PTT does not function. Since the satellite measures transmitter frequency, the stability and accuracy of the PTT carrier signal can effect results. Also, the degree of satellite coverage which varies according to latitude and is greatest at the poles and lowest at the equator will also affect the number of signals received. Consequently, a typical individual's track of locations can have large temporal and spatial gaps which leads to an irregular and unpredictable sampling rate. In addition to this, varying degrees of spatial error are associated with each of the reported location positions.

Failure to address these issues can severely impact further analysis (Hays et al. 2001; Bradshaw et al. 2007). Studies of area-restricted search (Fauchald & Tveraa 2003), behavioural mode (Tremblay et al. 2007) and home-range analysis (Austin et al. 2003) have all been shown to be detrimentally affected by spatial error (Costa et al. 2010). The size of the error, relative to the scale of measurement is an important factor. Studies that examine detailed aspects of

tracking data such as travel speed or small-scale movements are therefore more sensitive to these errors (Hays et al. 2001). The sampling frequency of Argos locations can also cause problems. While it is common to link time-series locations with a straight line (Tremblay et al. 2006), the larger the gap in the data, the less likely it is that the animal followed this path (Turchin 1998). Also, it is common practice to attach other bio-logging instruments in addition to location transmitters, e.g. time depth recorders (Burns & Castellini 1998; Bonadonna et al. 2000; Guinet et al. 2001). Since the sampling times of these data sets rarely align, studies that integrate locational data with additional telemetry data may require statistical techniques to interpolate intermediate positional data (Tremblay et al. 2006).

1.3 Data Handling

A number of options exist for obtaining location data from Argos including web site, ftp, email, etc. (Argos 2008). Regardless of access method, once received, these data need to be archived in a way that is secure and readily accessible when required for further analysis. Some options do currently exist including Argos' own system, ArgosWeb and the online application, Satellite Tracking and Analysis Tool (STAT) (Coyne & Godley 2005). These systems while being freely available and feature-rich, are intended primarily for the interpretation of location data using pre-defined analysis tools. As such, they are not specifically intended to aid researchers in the development of new analysis methodologies. In this thesis, a database designed specifically for incorporation into Argos location data analysis systems is presented.

1.4 Analysis of Argos Derived Location Data

Compared to data from more complex technologies such as the global positioning system, Argos locations are far less accurate and unpredictable (Patterson et al. 2010). Until recently, the only available data on the size of an individual location's error was a data field in the Argos record called a location class (Argos 2008). The location class categorised the potential error around a location as being within a range of values from < 250m through to an undefined level. This value refers to the radius of a circle and does not discriminate between

the longitudinal and latitudinal components of the error estimate (Argos 2008). Vincent et al. (2002) used a field based study to empirically determine error rates for each location class with separate estimates for each axis. Argos now supply a finer scale location specific error estimate in the form of an ellipse that allows for separate latitudinal and longitudinal error components (Argos 2008). Regardless of which method is used for informing the researcher of a location's uncertainty, the fact remains that these errors can have significant ramifications on the analysis and interpretation of these data. For instance, large enough errors can significantly misrepresent the actual position of an animal (Tremblay et al. 2009) and even small errors can result in an over-estimation of travel speed (Austin et al. 2003).

Most studies that deal with animal movement tracks of Argos location data employ some form of filtering in order to identify and remove locations with an excessively large error component. Typically, this is achieved by calculating inter-location travel speeds and comparing these to a pre-defined maximum speed. A commonly used technique for speed filtering is described in (McConnell et al. 1992). This method has been applied in numerous studies (e.g. Hays et al. 2001; Hull et al. 1997; Jonker & Bester 1998; Robinson et al. 2002; Sjoberg et al. 1995). It has also been used as the basis for more intricate filter methods as in the three stage filter of Austin et al. (2003) or the integration of speed, distance and turning angles as described in Freitas et al. (2008). These filters have been shown to discard fewer locations than the simpler method of McConnell et al. (1992). In many cases, however, each location was treated as if it were a precise fix thereby ignoring the error component and its potential impact on calculated speed.

Analysis of time-series location data that accounts for the uncertainty caused by location error requires innovative and computationally intensive algorithms (Jonsen et al. 2003). A new technique that is currently the subject of much research interest is state space modelling (SSM) (Jonsen et al. 2005; Patterson et al. 2008). This technique uses a combination of models such as observation error and individual behavioural states. These models are combined with knowledge of previous locations in order to predict an animal's future state. With a similar aim in mind, this thesis will describe an alternative approach to incorporating Argos error into a modelled distribution of spatial usage using a

search technique called extremal optimisation. One advantage of this technique is the ability to generate a result in less than a minute using currently available desktop computing.

A typical Argos animal track contains a set of locations that are irregularly sampled both spatially and temporally. This can cause large gaps in the data set that provide a great deal of uncertainty in the animal's intermediate movements (Freitas et al. 2008). Determination of spatial usage from these data is sometimes achieved through the application of kernel density estimation (KDE) (Worton 1989) which produces a two-dimensional probability distribution otherwise referred to as a utilisation distribution (Matthiopoulos 2003). The process of determining new data points given knowledge of existing data is known as interpolation (Lunardi 2009). When dealing with irregular sampled data such as that supplied by Argos, interpolation can be used to generate regularly sampled locations (Tremblay et al. 2006). This thesis will explore the application of KDE to Argos location data as well as the effect of incorporating interpolation methods with kernel smoothing algorithms for the generation of spatial usage maps.

1.5 Software

The development of software for the design and implementation of sophisticated analytical algorithms is a complex and time-consuming undertaking. Where analysis methodologies have inordinately high computational requirements, the choice of language and programming style can have significant effects on processing time and useability of the system. Many generic data analysis systems and languages exist, e.g. R, Matlab and Python. However, these languages are designed from a data manipulation perspective which is beneficial for effective design but not always computationally efficient. Lower level languages such C and C++ are harder to work with and maintain but can produce solutions that are orders of magnitude faster to run (Stroustrup 1997). These issues are addressed in detail within this thesis and various software library and application solutions are presented.

1.6 Thesis Outline

The overall aim of this thesis can be summarised in two parts. 1) To provide a concise set of tools for the analysis and spatial interpretation of Argos derived animal movement data. This includes the development of software for the storage and manipulation of spatial data as well as a generic framework for implementing complex multi-stage analysis algorithms. 2) The creation of new techniques for the analysis of these data. By quantitatively incorporating estimates of error and behaviour, these methods are designed with the intention of providing researchers with methods that preserve and interpret as much of the available information as possible. While the focus of this work was based on data acquired from an expedition to Heard Island, the techniques presented here are widely applicable to other sources of Argos location data. Although outside the scope of this thesis, specific aspects of this work could, with minor modification, be applied to other forms of spatial data such as light based geo-location (Field et al. 2004).

This thesis is structured as a series of studies, each of which address specific aspects of the manipulation and analysis of Argos derived animal location data. The order in which the chapters are presented is indicative of how the various ideas and methods evolved. Initially, the analysis system developed for the Heard Island expedition is described. This is followed by a more generic exploration of spatial usage analysis along with the integration of interpolation methods. Next, a new type of speed filter is presented and finally a detailed description of the underlying analysis software is given.

Chapters 2 – 5 were all written in the style of scientific papers with the intention of publishing in pier reviewed journals. There is therefore some repetition in concepts and methods. Wherever possible, findings from one chapter were cited rather than being duplicated but only where this would logically follow through to the citation format of stand alone papers.

Chapter 2: HeardMap

This chapter presents a complete software solution for the reception, storage, analysis and visualisation of Argos satellite location data. The main functions of this system were 1) retrieval of raw Argos data files through the

reception of Argos data emails; 2) a dedicated database system for the storage and manipulation of incoming location data along with track defining metadata and associated time-depth recorder (TDR) dive data; 3) analysis of spatial usage through the application of linearly interpolated pseudo-locations and kernel density estimation; 4) visualisation of location tracks and utilisation distributions with overlaid diving activity and 4D animation of time. This system was purpose designed and built for use in the field during the Heard Island predator-prey study that took place during the Austral summer of 2003/04. While not readily usable in other studies, it did highlight the need for sophisticated software tools and integrated analysis techniques for the storage and analysis of animal movement data. Attempting to interpret large and small scale movement patterns and spatial usage from remotely tracked marine animals poses many problems for researchers. A lot of the complexity in analysis begins with the irregular and unpredictable sampling rates of Argos locations combined with the potentially large spatial errors that can occur at each location fix. The rest of this thesis was designed in response to the work presented in this chapter. As such, the following chapters each address a specific issue relating to the analysis and interpretation of Argos location data.

Chapter 3: Kernel Density Estimation Techniques

Estimation of an animal's spatial usage is typically presented as a two-dimensional probability distribution known as a utilisation distribution (UD). The standard approach to generating a UD is through the application of a kernel density estimation (KDE) algorithm to a set of location data. A widely used method for KDE is described in Worton (1989). This technique was originally developed with radio tracking data and therefore requires location coordinates in Cartesian form. Since Argos location data is provided in geographical coordinates of latitude and longitude, utilisation of this method relies on a universal transverse Mercator projection of the location data in order to transform from Cartesian to geographical form. In this chapter I provide a modified version of KDE that allows for the direct input of geographic coordinates without prior transformation. Included in this work are geographic coordinate versions of weighted kernel smoothing and three forms of smoothing parameter estimation. Having defined the technique for smoothing, a validation methodology that tests

the predictive ability of various forms of UD is presented. Various combinations of weighted and unweighted KDE against different methods of smoothing parameter estimation are then quantitatively tested.

Chapter 4: Model Interpolated Kernel Smoothing

This chapter presents a new form of location interpolation known as model interpolated kernel smoothing (MIKS). MIKS uses a form of multi-objective evolutionary optimisation to iteratively search the space that an animal might travel through. The algorithm is able to integrate knowledge of location error distributions and population level distributions of inter-location travel speeds. The result is a set of pseudo-locations whose spatial distribution reflects the probability of the animal utilising a given region. By applying kernel smoothing techniques developed in the previous chapter, a UD is created from the pseudo-location data set. Validation tests between MIKS, linear interpolation, unweighted and weighted versions of KDE are then performed so as to evaluate the performance of this approach. This chapter provides a natural extension to the methods presented in chapter 3. These chapters are therefore intended to be published as a two part paper.

Chapter 5: Filtering Location Data

All studies that make use of Argos location data must first address the issue of location error. While some degree of spatial error exists for all locations in a track, a minority of locations in a typical track will possess an error large enough to make that location highly improbable. Although some studies do attempt to account for the location error term variance as part of a larger statistical analysis, most papers still rely on destructive filtering methods to identify and remove these outlier locations. Of these methods, the most widely used is the speed filter described in McConnell et al. (1992). This method, although simple to implement can result in an excessive number of locations removed. The technique described in this chapter uses a modified version of the optimisation algorithm described in chapter 4 to find a set of locations that pass through Argos location regions. These regions are defined by the size of the estimated error about each reported location. The output of this filter is a set of adjusted locations with some locations removed. The performance of this filter was tested against that of

McConnell et al. (1992) when run across several tracks from multiple seal and penguin species.

Chapter 6: Software Design

The various algorithms and data analysis described in previous chapters prompted the design and development of a sophisticated, modular and extendible software platform. This system, known as Track, was written using high performance languages and programming styles and includes a dedicated tracking database. This chapter describes the design and use of this system in the context of the methods presented in the body of this thesis.

Chapter 7: General Discussion

The final chapter in this thesis summarises the application and assessment of the various analytical techniques that were developed. The work is presented in the context of existing solutions and several options for further research are suggested.

Chapter 2

HeardMap - Tracking Marine Vertebrate Populations in Near Real Time[1]

Abstract

I developed a computer program (HeardMap) to receive, manage, visualise and analyse Argos location data and synchronised archival dive data. This program was used to acquire a relative intensity scale of areas of foraging and non-foraging activity of marine vertebrate predator populations around Heard Island in near real time during the austral summer of 2003/2004. Foraging and non-foraging areas were determined by using HeardMap and trawl and acoustic samples were collected using the research ship Aurora Australis to measure the prey characteristics in those areas. A total of 226 meso-pelagic predators were tagged on Heard Island over a two month period. Most of these animals foraged in an area to the east of the island. The use of HeardMap demonstrated that large numbers of predators of various species could be tracked simultaneously and near real time estimates of foraging intensity could be derived. The inclusion of dive data in the data display provided a measure of behaviour embedded in the spatial distribution of the animals.

2.1 Introduction

With the advent of modern telemetry and data-logging technologies, scientists are able to monitor the movements and behaviour of large marine animals at sea with increasing precision. This has been facilitated by several factors including lower price and size and the increasing sophistication and availability of tags. As a result, much larger volumes of data can now be derived from tagging programs due to increasing sample size, deployment duration and sampling frequencies (Boveng et al. 1996; Eckert & Stewart 2001; Fedak et al. 2001).

[1] Frydman, S. and Gales, N. (2007) HeardMap: Tracking marine vertebrate populations in near real time. Deep Sea Research Part II: Topical Studies in Oceanography 54, 384–391.

A large-scale marine ecosystem study at Heard Island in 2003-2004 aimed to determine the relationships between oceanography, primary and secondary production (chlorophyll, zooplankton), mesopelagic biota (fish and squid) and land-based predators (penguins and seals). The study incorporated an adaptive research design that used predator track and diving information in near real-time to determine the locations for sampling potential prey of predators as close to the time as possible when predators were foraging in those areas. Samples in areas where tagged predators did not forage were also sampled. The results of this work are being used to evaluate the availability of different kinds of prey to seals and penguins based at Heard Island.

The study involved the coordination of research teams based at two camps on Heard Island with a another team on board the research vessel, *Aurora Australis*. Satellite radio tracking tags were deployed on Macaroni penguins (*Eudyptes chrysolophus*) and King penguins (*Aptenodytes patagonicus*) and on Antarctic fur seals (*Arctocephalus gazella*) along with Time Depth Recorders (TDR) to log diving behaviour.

In order for the ecosystem study to be successful, an integrated software application was needed that could handle all aspects of data acquisition, storage, manipulation, visualisation and analysis of data obtained from track and diving recorders deployed on seals and penguins. This system needed to function in as close to real time as possible. The aim of this paper is to describe the development of a system that meets these diverse requirements (HeardMap) and to demonstrate how this program was used to successfully navigate the Aurora Australis to sample locations.

2.2 Methods

Location data was provided by attaching Argos satellite transmitters (Platform Transmitting Terminal; PTT) to the study animals. Argos is a satellite based system where locations of animal-borne PTTs are estimated using calculations of changes in transmitter signal frequency. These changes are due to the Doppler shift caused by the movement of the satellite receiver relative to the transmitter. When a satellite passes over a ground based listening station, its data are downloaded, processed and electronically available to users within minutes to

hours of detection Argos, 2008, #76523}. Data for our study were transmitted from the French ground station every four hours to the Australian Antarctic Division (AAD), Tasmania, Australia. These data were then packaged up by an automated email delivery system and delivered to both the field parties on Heard Island and the Aurora Australis operating nearby via a satellite phone with a modem attached. The location data emails were then read and parsed by the HeardMap program and the data stored in its database.

In order for HeardMap to manage the entire process from data reception to foraging density analysis (from which the ship transect design could be determined) it incorporated the following functions (summarised in Figure **2.1**).

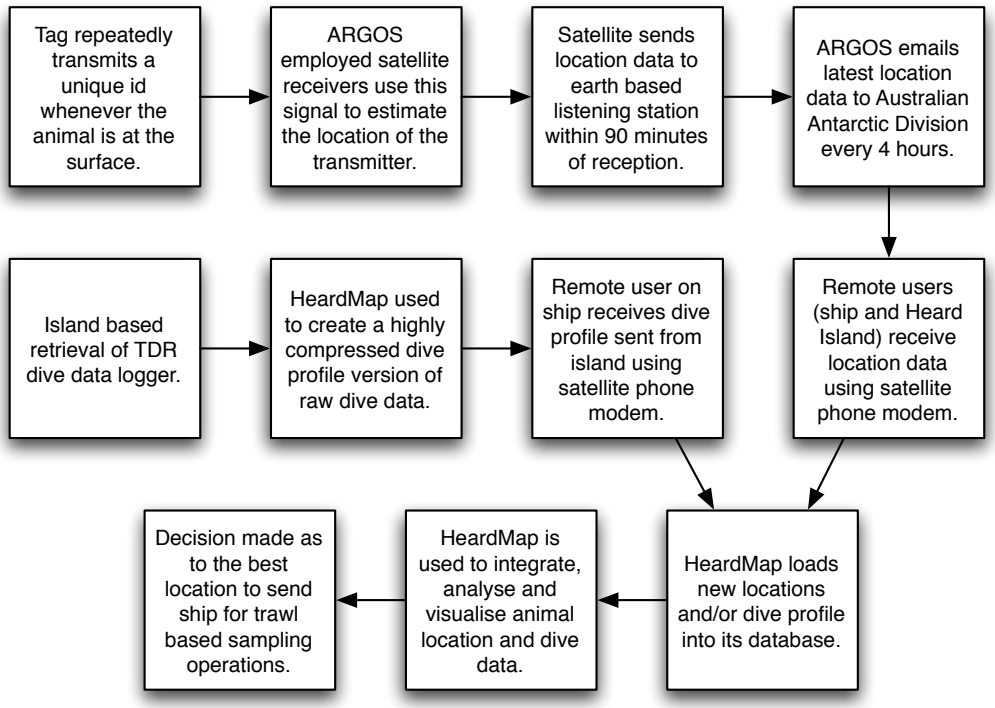

Figure 2.1. *The sequence of data flow from acquisition through to transmission, processing and analysis. This diagram illustrates how the HeardMap system was used to centralise the management and analysis of a logistically and technically complex experiment.*

2.2.1 Data parsing and storage of Argos data

Raw data from the Argos system was received via email as a text message in a proprietary format (Diag format) (Argos 2008). These files were parsed and the data stored in a SQL-compliant database (Associates 2003).

2.2.2 Compression and transmission of TDR data

Comma delimited text files were obtained from Instrument Helper, a software package supplied by the tag manufacturer (Wildlife Computers 2003) and used to download and decode the data stored on the TDR tags. These files were read by HeardMap which then generated and saved a dive summary. These results needed to be transmitted from the Aurora Australis to Heard Island across a slow and expensive satellite phone link. In order to save transmission time, a dive summary was created by reading each depth value in the dive record but only recording the value at each inflection point. These inflection points which denote maxima and minima in the dive summary were found by reading each depth recording in sequence and determining the position where the change in depth shifted from positive to negative or negative to positive. These summary data were then stored in an XML format and zip compressed. This resulted in the diving activity of the animal being preserved at a resolution that was adequate for our purposes while reducing the file size to approximately 1 percent of the size of the original.

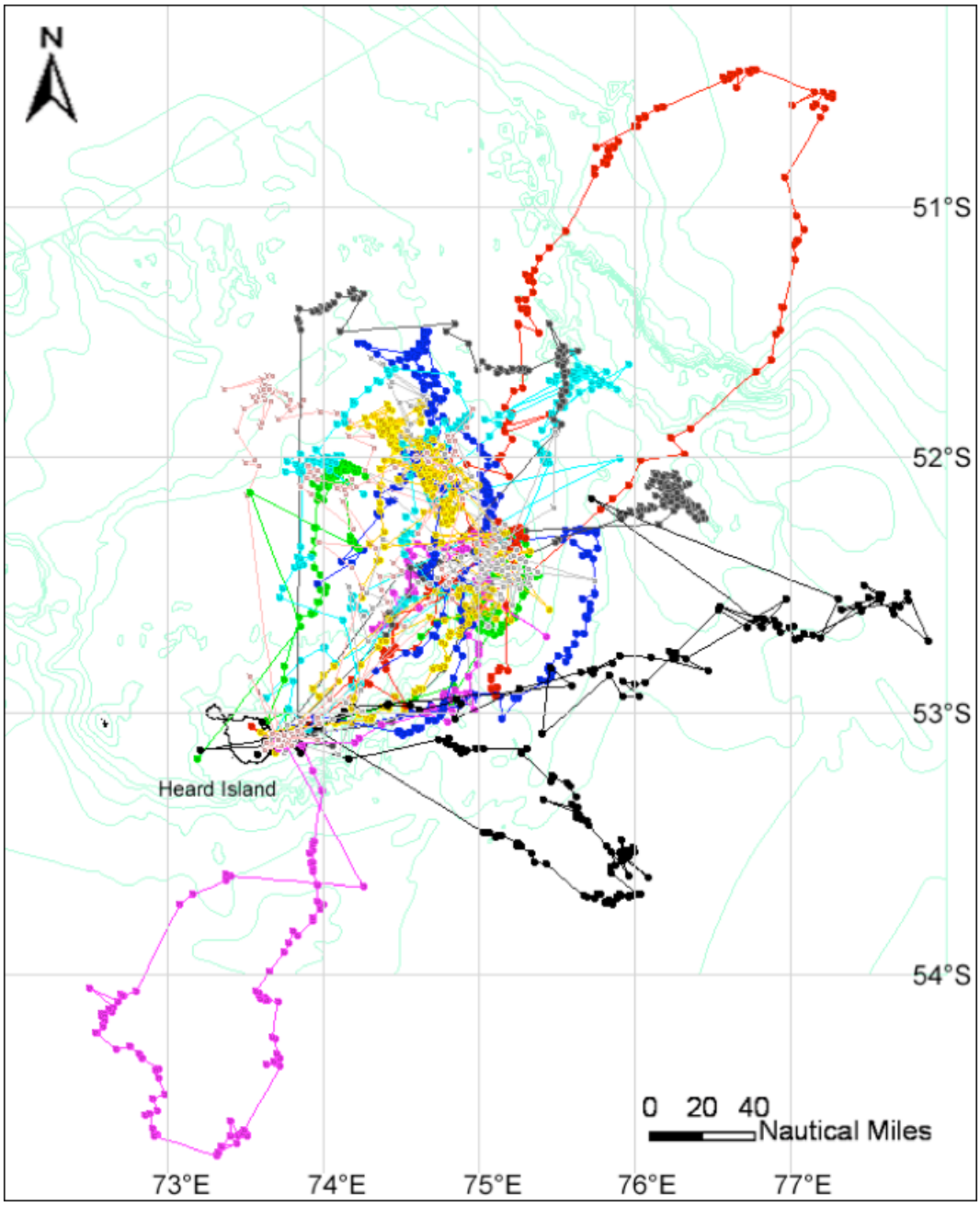

Figure 2.2. *Female Fur Seal Tracks around Heard. This plot shows the tracks of 20 individuals for a single foraging trip each between the 20 December 2003 and 11 January 2004. A total of 2309 velocity filtered locations were used to construct this plot. Dots represent the Argos locations with the temporal sequence shown by linking the dots with straight lines.*

2.2.3 Bout Detection and geo-location

A dive was deemed to begin when the dive depth exceeded 4 metres and to end when the depth subsequently returned above 4 metres. Individual dives were then grouped together into bouts using a simple algorithm. The beginning and end dives for a bout were determined by finding dives where the preceding surface interval (in the case of the start) and subsequent surface interval (for the

end) exceeded 1 hour. Geo-referencing of dive bouts was achieved by taking the times for the beginning and end of each bout and matching those times to the Argos-derived location of the animal.

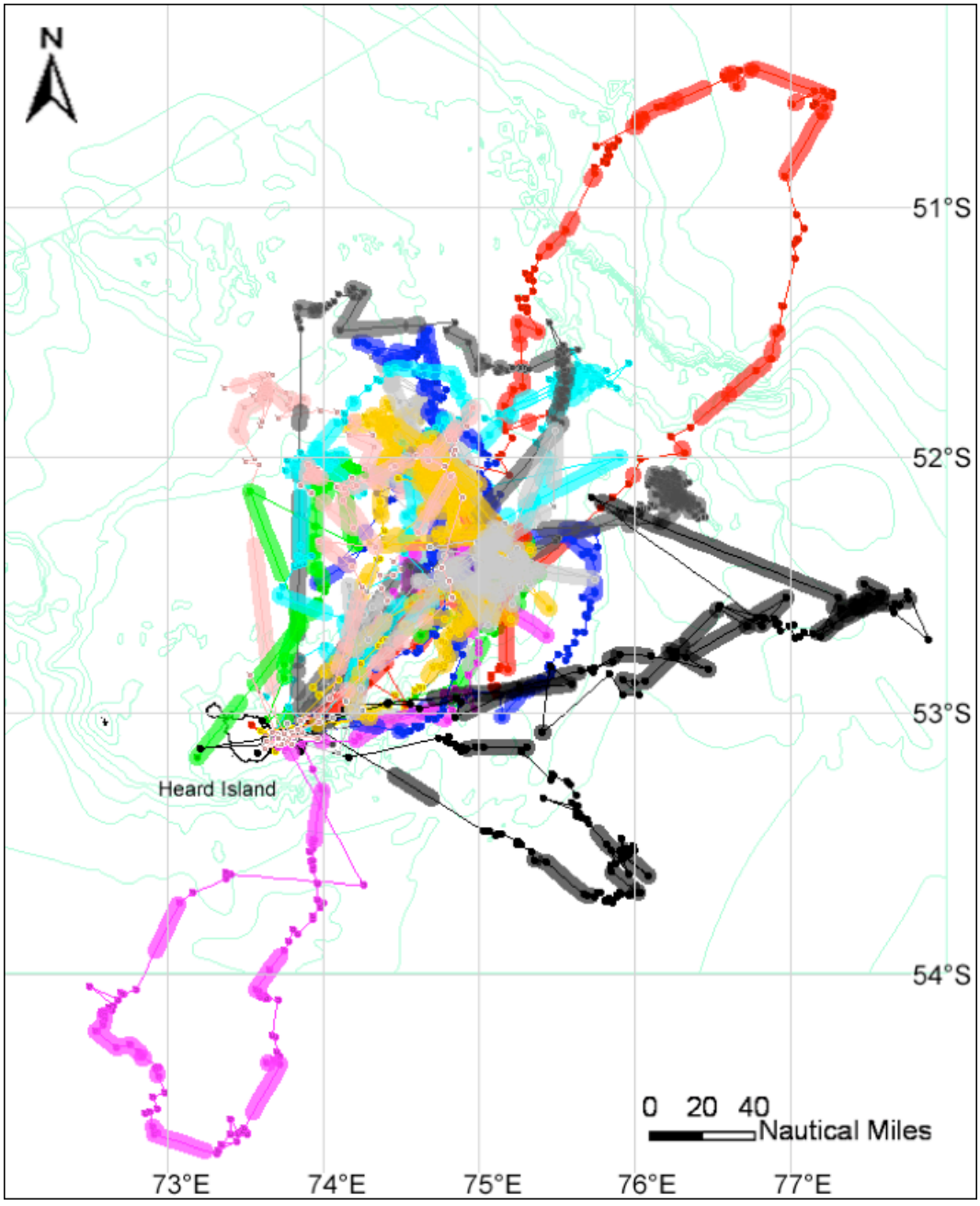

Figure 2.3. *Geo-located dive bouts overlaid on tracks displayed in Figure* ***2.2****. Wider, slightly transparent lines represent the location of dive bouts. The actual diving that took place within a bout is viewable by clicking on that bout.*

2.2.4 Database management

HeardMap incorporates a multi-table relational database system (Associates 2003). A database management tool was written and included in the main

program. This provides a mechanism for setting up track and dive metadata within the database and for grouping dives.

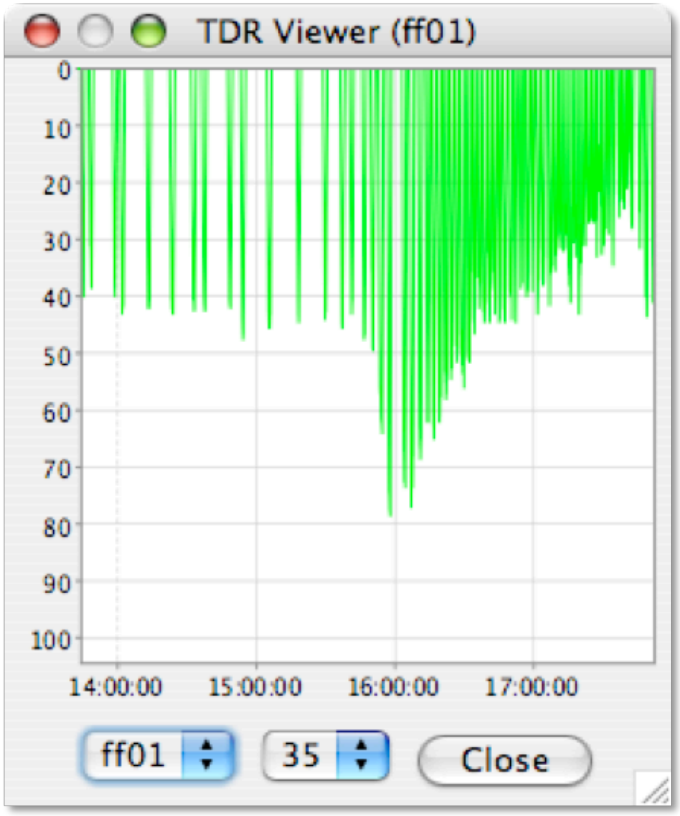

Figure 2.4. *Example of detailed dive screen that appears when a dive bout from Figure* ***2.3****. was clicked on using the mouse. Depth in metres is shown on the vertical axis and time on the horizontal axis. The drop down boxes allow for the selection of a different animal and a different bout respectively.*

2.2.5 Velocity Filtering

Argos supplies a class rating of location with each datum (location) where class Z is regarded as invalid. We therefore utilised all classes except for Z. Argos location data often includes an unacceptably large error component (Vincent et al. 2002; Thompson et al. 2003b). It is therefore necessary to pre-process these data before further analysis. A commonly used technique (Arnould & Hindell 2001; Campagna et al. 2001; Lesage et al. 2004) is to apply a velocity filter that uses an iterative root mean square algorithm to remove erroneous locations (McConnell et al. 1992). A location is taken to be in error and discarded when its inclusion causes a section of the track to exceed a given maximum velocity parameter. We set the maximum velocity for Antarctic Fur seals and King penguins to 20 km/h and for Macaroni penguins we set it to 15 km/hr. Here, we introduce a minor modification to this method. The Diag format file supplied

by Argos, always contains an alternate location. This is a product of the Doppler shift algorithm employed by Argos and is usually invalid (Argos 2008). However, on approximately one percent of occasions the alternate location is the correct location to use. In order to retain this information, we first run the velocity filter using only the first location set. We then substitute in the alternate locations for each location that failed the first pass of the filter. We then run the filter again. All further analysis is performed on the output from this second pass.

2.2.6 Visualisation

Geographic visualisation was achieved using an open source GIS package (BBN Technologies 2003) that was incorporated into the main HeardMap application. Shapefile layers of the island perimeter and surrounding bathymetry were overlaid with dynamic specialised layers that could display information such as Argos locations, interpolated locations of diving activity and an area usage grid. Velocity filtered Argos locations (see section 2.2.5) were placed on the map as small dots with straight lines connecting them (Figure **2.2**). Dive bouts were displayed on the map as a thicker line around the location layer lines (Figure **2.3**). The positioning of the bout line was determined by the start and stop time of the bout. The actual diving activity that occurred within each bout was then viewable by using the mouse to click on the dive bout line. This caused a separate window to appear that contained the more detailed view of individual dives (Figure **2.4**). The area usage grid was generated by adding pseudo-locations in 30 min increments between each pair of consecutive locations in a track. The actual location of each pseudo-location was determined by a linear interpolation between the two actual locations where the position along this line was a proportion of the time difference between the two actual locations. This process was applied to all tracks within a group. As this form of interpolation imposes an assumption of straight line travel between known locations, kernel smoothing was used as a way of spreading this effect out thereby allowing for likely deviations from the straight line track. The kernel smoothing function was applied using the equation

$$KS = \frac{1}{2\pi n h^2} \sum_{i=1}^{n} \exp\left(\frac{d^2}{2h^2}\right) \quad (1)$$

where h is the smoothing parameter and d is the distance of the ith observation from the closest grid cell centre (Worton 1995). Distances were calculated in

kilometres and *h* was set via experimentation to 10 as this was found to give a good qualitative display. This produced a two dimensional grid that could be overlaid on the map. The grid cell colouring was determined by first rescaling the grid cell values to range from 0 to 255 where 0 equated to the lowest value and 255, the highest. The rescaled values were then used to obtain colours from a colour look up table that referenced a gradient from blue to red (Figure **2.5**). Clearly, this process meant a loss of absolute value information, however this was deemed acceptable as we were only interested in the relative values of time usage across the geographical range of animal movement. Finally an adjustable time window was added, allowing the user to limit the view to a specific time and date range. By moving a slider, it was possible to animate usage through time.

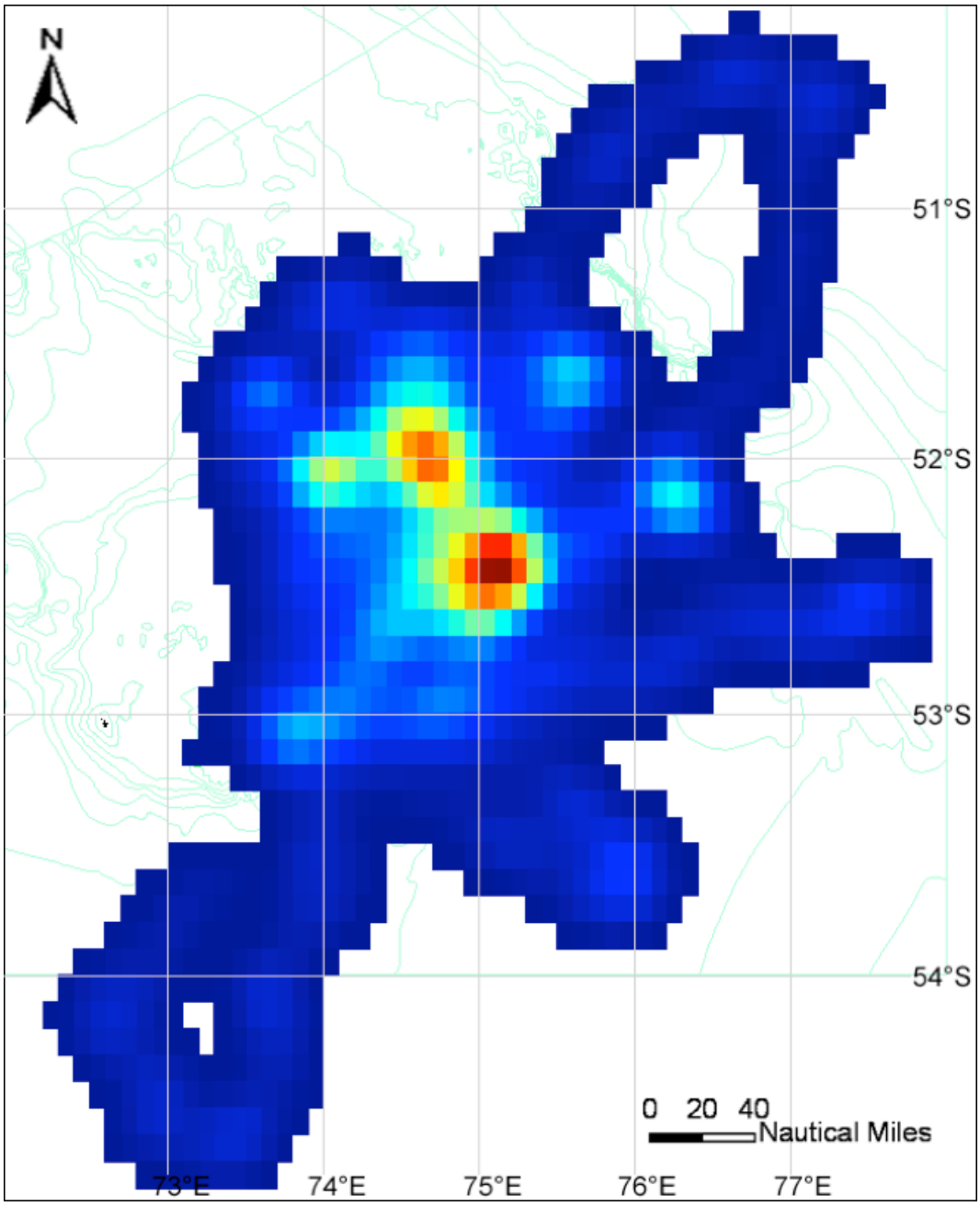

Figure 2.5. *Area usage grid of the tracks displayed in Figure* ***2.2****. The relative amount of time that a number of animals spent in a given area is represented by the colour where blue is low and red is high.*

The process for determining trawl locations involved an assessment of the activity of foraging predators using the information provided by the area usage grid. The experiment was designed to sample a foraging and non-foraging region for each of the three study species. The decision as to where next to trawl was made by using HeardMap to display the tagged animals' recent activity. Where this activity was highest was deemed to be a foraging area and where it was lowest, a non-foraging area. As new data was continually being received, the data

analysis and subsequent trawl location could be reviewed and if necessary modified, right up until the time a trawl commenced.

2.3. Results

Animals were tracked from 18 December 2003 through to 29 January 2004 with instruments being retrieved and redeployed on different individuals at the end of each foraging trip. The numbers of individuals tracked were 50 female Antarctic Fur seals, 20 male Antarctic Fur seals, 50 King penguins and 106 Macaroni penguins. The majority of tagged animals spent their time at sea to the east of the island. Trawl sampling operations on the Aurora Australis took place between 23 December 2003 and 23 January 2004 and were concentrated in the main areas where animals foraged.

The movement of the various tagged animals was visualised in a number of ways. An overview of velocity filtered Argos locations was obtained by plotting just the locations joined by lines. This was usually done on a per species basis and can be seen for the female Antarctic Fur seals in Figure **2.2**. In order to factor in the time component the area usage grid was then added (Figure **2.5**). Finally, the location and structure of dive bouts were displayed on a group basis (Figure **2.3**) and in greater detail by clicking on individual bouts (Figure **2.4**).

2.4. Discussion

The use of HeardMap during the Heard Island study provided a group of geographically separated researchers with a reliable and robust tool for the storage, management, analysis and visualisation of a large and continuously changing dataset. The integration of a number of technologies and statistical algorithms into a single streamlined package enabled *in situ* adjustments to the experimental design to be made using information that was at most only a few hours old. This in turn meant that the information provided by multiple simultaneously deployed satellite tags could be used to promptly navigate the *Aurora Australis* to areas of interest. This was vital to the success of the experiment as the ability to relate observations of availability of biota using trawls to the foraging activity of predators would diminish as the time delay between tracking and trawling increased.

The area usage grid was created to provide a scaled visual display of relative levels of time spent in a given area. This was based on the premise that extended periods of time in a given area should correlate well with increased foraging activity. By viewing the geo-referenced dive data overlayed on the usage grid, we were able to confirm that increased time spent in a given area was indicative of an increase in diving and therefore foraging activity. Area usage can be estimated by applying a kernel smoothing function (Simonoff 1996) to a set of geographical locations and reading the areas of higher density as increased usage (e.g. Amstrup et al. 2004; Hedd et al. 2001; Phillips et al. 2004). The problem with this approach is that if the time between location acquisitions is inconsistent (as is the case with satellite derived animal tracking data), then the smoothing product reflects only densities of where locations were measured and does not account for the actual time spent in an area. For instance, multiple locations in the one area could be indicative of several minutes, hours or days of activity. As the likelihood and quality of location is determined by the behaviour of the animal at the surface, then different behaviours such as diving, resting and travelling are likely to result in variable location acquisition rates. We therefore developed a new method for area usage estimation that involves the use of interpolated pseudo data which enabled each datum to be weighted by a function of time between successive data points. This gave a more accurate representation of potential time spent in a given area than would have been possible with kernel smoothing alone.

A further substantial advantage of HeardMap was its utility for exploring the large and continuously updating dataset. In particular the capacity to set a brief time window (perhaps 12hrs) and to slide this through the preceding days or weeks of the experimental period produced a clear visualisation of the relative heterogeneity of areas of foraging intensity which could be stratified by species, sex or individual animal.

The development and application of HeardMap illustrates the benefit of a user friendly software package that can handle complex tracking datasets for research scientists to use both in the field and back in the lab. While HeardMap itself was developed specifically for this study, much of its functionality is being moved across to a new system that will allow for a more flexible, modular

approach to data handling and analysis. This new system, titled Track, is currently in development and will be freely available in the near future.

Chapter 3

Kernel Density Estimation Techniques for the Generation of Utilisation Distributions from Geographic Coordinate Location Data

Abstract

A marine animal's home range and utilisation distribution is commonly determined through the application of kernel density estimation (KDE) to a set of satellite derived location data. The requirement for Cartesian coordinate data in the standard set of KDE calculations means that Argos derived geographic coordinates (latitude and longitude) must first be transformed with an appropriate map projection. Further considerations in the use of KDE are the choice of an unweighted or weighted kernel and the method for determining the level of smoothing. This chapter presents the equations necessary to generate KDEs directly from geographic coordinates. All necessary calculations for weighted kernels and the three most commonly used techniques for determining the smoothing parameter are also modified to use this coordinate system. Example UD maps developed with unweighted and time-weighted KDEs and combined with the different smoothing parameter techniques are given. Multiple validation tests that measure the overlap of UDs generated from track data subsets were used to test the performance and consistency of the various forms of KDE. Significance tests and plots showed that the most appropriate form of KDE for Argos derived tracking data was a time-weighted kernel with a smoothing parameter determined by likelihood cross-validation.

3.1 Introduction

The home range of an animal is defined as the area that an individual traverses during the course of its normal activities (Burt 1943). Detailed knowledge of an animal's movements within its home range can provide valuable behavioural and ecological insights (Horne & Garton 2006b). The usual method

for quantifying an individual's home range is to generate a two-dimensional probability density function otherwise known as a utilization distribution (UD) (Worton 1989). When information on an animal's movements is limited to occasional positional samples, statistical techniques are needed to estimate the animal's UD (Hines et al. 2005). A commonly used method for determining a UD is to calculate a kernel density estimate (KDE) using a standard bivariate normal density kernel (Worton 1995; Blundell et al. 2001; Vokoun 2003; Austin et al. 2004). By including a weighting parameter, the relative influence of each observation to the structure of the KDE can be adjusted. A weighted KDE can be used to incorporate additional information such as the effects of temporal and spatial correlation (Katajisto & Moilanen 2006) or to compensate for the irregular temporal sampling of location data (Thompson et al. 2003a).

The kernel smoothing method defined in (Worton 1995) assumes that location data and grid cell positions be supplied as Cartesian coordinates (Wood et al. 2000). Since Argos locations are always supplied as geographic coordinates (Argos 2008), it is necessary to first transform these locations into Cartesian form, e.g. through the application of a universal transverse Mercator projection (Keating & Cherry 2009). By modifying the KDE calculations from Euclidean distance measures to great circle distances, the need for transformation between coordinate systems can be avoided. This simplifies the use of a KDE when applied to geographical data and has the added benefit of not requiring a reverse transformation where the resulting UD is to be interpreted in geographic coordinates.

A critical consideration in the application of kernel smoothing is the selection of an appropriate value for the smoothing parameter, h (Silverman 1986). The size of h determines the amount of smoothing such that a large value for h reveals the general shape of the distribution while a smaller value shows more of the fine scale detail (Seaman & Powell 1996). Adaptive kernels that vary their smoothing value in response to the density of local observations have also been proposed (Silverman 1986; Worton 1989). However, simulation results have found that a fixed kernel, where the smoothing value is constant, is a better home range estimator (Seaman et al. 1999). Methods for determining the smoothing value of a fixed kernel include manual selection (Seaman et al. 1998),

the h_{ref} method (Worton 1995), least squares cross-validation (h_{LSCV}) (Worton 1995) and likelihood cross-validation (h_{LHCV}) (Matthiopoulos 2003). The h_{ref} method has been shown to produce good estimates when applied to a unimodal distribution. However, when used on clumped data as is often found in animal tracking locations, it has a tendency to over smooth (Worton 1995). The h_{LSCV} and h_{LHCV} cross-validation techniques both use the h_{ref} value as their starting point and then seek to minimise their respective functions across a range of h values either side of h_{ref}. h_{LSCV} has been used in a few studies (Matthiopoulos 2003; Horne et al. 2007) and has been shown to perform better than LSCV especially on small sample sizes of less than 50 observations (Horne & Garton 2006a). In spite of this, h_{LHCV}, which was originally recommended by Worton (1995), continues to be the most frequently used method (Horne & Garton 2006b).

Validation of the accuracy of a UD derived from sampled data is complicated by the lack of knowledge of the true UD. In such cases, the accepted technique is to subdivide the data into two separate data sets, one for estimation and the other for validation (Matthiopoulos 2003). By generating a UD for each subset and determining the goodness of fit between the two distributions, a quantitative measure of the ability of the UD to predict its complimentary UD is produced. In this chapter I present a series of equations that allow for the creation of weighted and non-weighted KDEs directly from geographic coordinate data. I also develop the method for estimating the smoothing parameters h_{ref}, h_{LSCV} and h_{LHCV} from the same data. Using two example tracks of Antarctic fur seal location data, I describe a technique for validation that quantitatively evaluates the predictive ability of the different UDs. I then apply this to a full factorial analysis of the weighted and non-weighted KDEs combined with the smoothing estimation methods, h_{LSCV} and h_{LHCV}.

3.2 Methods

3.2.1 Data Collection

The location data used in this study were obtained from the Antarctic fur seal (*Arctocephalus gazella*) colony at Spit Bay, Heard Island (53° 07′ S, 73° 44′ E) between December 2003 and February 2004. All data was received using the Argos satellite system. Platform terminal transmitters (PTT) manufactured by

Sirtrack Limited were attached to individual female fur seals. The PTT was glued to the back of the animal using Araldite 2017 epoxy adhesive (Vantico, Switzerland). The PTT was set to transmit on a continuous duty cycle with a 30 second repetition rate (Robinson et al. In Prep).

3.2.2 Distance and Time Calculations

All distances and speeds were calculated using spherical trigonometry (Zwillinger 2003; Freitas et al. 2008) such that:

$$D_{i,j} = GCD\left(lat_i, lon_i, lat_j, lon_j\right) = \operatorname{acos}\begin{pmatrix}\sin(lat_i) \times \sin\left(lat_j\right) + \cos\left(lat_j\right) \\ \times \cos\left(lat_j\right) \times \cos\left(lon_j - lon_i\right)\end{pmatrix} \times C \quad (3.1)$$

where $C = \frac{60}{1852} \times \frac{180}{\pi}$ and distance $D_{i,j}$, between locations i and j, is given in metres. Angles are in radians. Velocity,

$$v_{i,j} = \frac{D_{i,j}}{T_j - T_i} \quad (3.2)$$

is in metres/second and $T_j - T_i$ is the time in seconds between locations i and j.

Individual locations, L_i, were assigned a time duration TD_i as a consequence of the acquisition times of the immediately adjacent locations such that,

$$TD_i = \frac{T_i - T_{i-1}}{2} + \frac{T_{i+1} - T_i}{2} \quad (3.3)$$

3.2.3 Speed Filter and Haul Out Exclusion

This filter was implemented as described in McConnell et al. (1992). It is an iterative algorithm that repeatedly calculates the velocity, v_i between adjacent locations. After calculating all location velocities, the fastest location is removed if it exceeds the given maximum speed. This process is repeated until all velocities fall below the maximum speed. The formula used to calculate velocity is:

$$V_i = \sqrt{\frac{1}{4} \times \sum_{j=-2, j\neq 0}^{j=2} \left(v_{i,j+i}\right)^2} \quad (3.4)$$

Time spent at the animal's haul out site either before or after an at-sea foraging trip was detected and removed by excluding all locations that fall within

a circle of radius 5 km and centred at the haul out site. The actual travel to and from the haul out site was included by not excluding the last location to be recorded within the circle radius before the track begins and the first location in that circle that follows the end of the track.

3.2.4 Weighted Statistics

Where a statistic was modified to incorporate weighted observations, the sample size n, was replaced with the weight adjusted sample size according to

$$n_w = \sum_{i=1}^{n} w_i \tag{3.5}$$

where w_i is the individual observation weight. In the case where all weights are set to 1 then $n_w = n$ and the weighted equations default back to their unweighted form. The weighted mean was calculated as

$$\bar{x}_w = \frac{\sum_{i=1}^{n}(w_i \times x_i)}{n_w} \tag{3.6}$$

and the weighted sample variance as defined by Galassi et al. (2009) using

$$s_w^2 = \frac{\sum_{i=1}^{n} w_i}{\left(\sum_{i=1}^{n} w_i\right)^2 - \sum_{i=1}^{n} w_i^2} \sum_{i=1}^{n} w_i (x_i - \bar{x}_w)^2 \tag{3.7}$$

3.2.5 Weighted Kernel Density Estimation

The equation for calculating a weighted KDE grid cell is given by

$$wKDE(X) = \frac{1}{2\pi n_w h^2} \sum_{i=1}^{n} \left(w_i \times \exp\left(-\frac{D_{i,X}^2}{2h^2} \right) \right) \tag{3.8}$$

where h is the smoothing parameter, D_i is the distance from the ith location to a position, X on the sphere and w_i is the weighting for the ith observation. A utilization distribution is then constructed by evaluating this function for a given value of h over a grid of X coordinates.

3.2.6 Smoothing Parameter Selection

3.2.6.1 Weighted Reference Method

The reference method for determining h is defined by (Worton 1989) as

$$h_{ref} = \sqrt{\frac{s_x^2 + s_y^2}{2}} \times n^{-1/6} \tag{3.9}$$

where n is the number of locations and s_x^2 and s_y^2 are the respective sample variances of the x and y components of the Cartesian coordinate locations.

A weighting component and direct use of geographic coordinates was incorporated into Equation 3.9 such that

$$h_{wref} = \sqrt{\frac{s_{wdlat}^2 + s_{wdlon}^2}{2}} \times {n_w}^{-1/6} \tag{3.10}$$

where s_{wdlat}^2 and s_{wdlon}^2 are the respective weighted sample variances (equation 3.7) of the longitudinal and latitudinal location displacements from their means as determined by great circle distance. The latitudinal and longitudinal displacements of the *i*th location were determined relative to the latitudinal and longitudinal means according to

$$dlat_i = GCD\left(lat_i, \overline{lon}, \overline{lat}, \overline{lon}\right) \tag{3.11}$$

$$dlon_i = GCD\left(\overline{lat}, lon_i, \overline{lat}, \overline{lon}\right) \tag{3.12}$$

3.2.6.2 Weighted Cross-validation

Determination of h by cross-validation is achieved by finding the value of h that minimises either the function $LSCV(h)$ in the case of least squares or $LHCV(h)$ if using the likelihood method (Horne & Garton 2006a). Typically the range of h to test is defined as $0.1h_{ref} < h < 1.5h_{ref}$ (Worton 1995). Calenge (2007) provides a function in R (Ripley 2001) called kernelUD() that has an option for minimising $LSCV(h)$ (Horne & Garton 2006a). This function works by dividing the range $0.1h_{ref} < h < 1.5h_{ref}$ into 100 evenly spaced steps and selecting the smallest value of $LSCV(h)$ after the function has been run 100 times. Here, I use a more efficient approach based on a ternary search (Ajay 2010). Ternary search works by recursively dividing the range of h values into thirds. The bounds of each section are then compared to find if the minimum resides in the bottom or top two thirds before repeating the ternary divisions on the selected smaller section until the minimum is found. Using ternary search required only 10 evaluations of the cross-validation function. The function to minimise for weighted LSCV is given by

$$wLSCV(h) = \frac{1}{\pi h^2 n_w} + \frac{1}{4\pi h^2 n_w^2} \times \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\left(\exp\left(-\frac{D_{ij}^2}{4h^2}\right) - 4\exp\left(-\frac{D_{ij}^2}{2h^2}\right)\right)\times w_j\right) \quad (3.13)$$

Weighted LHCV was found by minimising

$$wLHCV(h) = -\frac{1}{n_w} \times \sum_{i=1}^{n} \log(wKDE_{-i}(L_i)) \quad (3.14)$$

where $wKDE_{-i}(L_i)$ is the weighted KDE of the *i*th location, L_i relative to all other locations excluding itself. For both forms of cross-validation the range of h to minimise was based on the weighted reference method such that $0.1h_{wref} < h < 1.5h_{wref}$.

3.2.7 Generating the Utilisation Distribution

After selecting an appropriate value for h, the wKDE is applied repeatedly across a rectangular grid of cells that is bounded by the latitudinal and longitudinal range of the set of track locations. The size of the grid was extended by 10000 metres in all directions so as to fit the entire UD which as a result of smoothing extends beyond the range of the input locations. A cell size of 2000m x 2000m was chosen experimentally as a reasonable trade-off between computational requirements and level of detail.

Where a grid cell is a large distance from any track location, its value is negligible compared to the cell values in the true distribution surrounding the locations. Since these small but non-zero values were likely to be a product of the *wKDE* function rather than a true measure of probability. Therefore, all values less than 1% of the UD's maximum cell value were removed such that

$$UD_i' = \begin{cases} 0 \; if \; UD_i < \max(UD) \cdot 0.01 \\ UD_i \end{cases} \quad (3.15)$$

The final step in creating a probability density function was to normalise the remaining grid values so that the grid values summed to 1 by applying

$$UD_i'' = \frac{UD_i'}{\sum_{j=1}^{n_{UD}} UD_j'} \quad (3.16)$$

where n_{UD} is the number of cells in the grid.

3.2.8 Comparing Utilization Distributions

The quantitative difference between UDs was calculated using the coefficient of variation of the root mean square error (Van der Weken et al. 2003; Zar 2009). This statistic is used to give a measure of precision of one UD's prediction of the other, i.e. the degree of scatter of the data in one UD relative to the other (Kissock & Seryak 2004). The equation is given by

$$\mathrm{CV(RMSE)} = \frac{\sqrt{\frac{\sum_{i=1}^{n_{UD}}(UD1_i - UD2_i)^2}{n_{1\cup 2}}}}{(n_1^{-1} + n_2^{-1})/2} \tag{3.17}$$

where n_{UD} is the total number of grid cells in the UD and n_1 and n_2 are the number of cells in $UD1$ and $UD2$ respectively that have a value greater than zero. $n_{1\cup 2}$ is the count in either UD where for each cell position, i, at least one of the two cells has a value greater than zero.

Conversely, the similarity of the complementary UDs was determined using Bhattacharyya's affinity (Thacker et al. 1997) whose value increases in response to the degree of overlap of the two distributions (Fieberg 2007). The value of BA ranges from 0 for no similarity up to 1 for an exact match. It is calculated using,

$$BA = \sum_{i=1}^{n_{UD}} \sqrt{UD1_i \times UD2_i} \tag{3.18}$$

where n_{UD} is the total number of grid cells.

3.2.8 Validation of Utilisation Estimates

The track location data set was randomly separated into two equally sized subsets. Unweighted and weighted UDs were generated using both h_{LSCV} and h_{LHCV} smoothing parameters for each of the paired subsets. Time duration, TD_i (Equation 3.3) was used to set the value of the individual location weight, w_i in the weighted kernel estimator, $wKDE(X)$ (Equation 3.8). The size, position and grid cell count of the subset UDs was determined from the complete set of data so as to ensure that the cell locations of the subsets overlapped exactly. The difference and similarity of the paired UDs was measured using $CV(RMSE)$ (Equation 3.17) and BA (Equation 3.18). The reproducibility of these metrics when applied to differing random subsets was determined by repeating this

process 100 times for each track. The raw results of the 100 tests for CV(RMSE) and BA were plotted using simple line plots. The paired nature of the unweighted versus weighted results was accounted for by using parallel line plots (McNeil 1992). Significance tests of unweighted versus weighted measures of CV(RMSE) and BA were made using paired t-tests and a two-way repeated measures analysis of variance (using ezANOVA in the R package ez) (Lawrence 2011).

Foraging trip data from two female Antarctic fur seals were used to test the predictive power of the subset data UDs. Tracks were selected that highlighted different patterns of foraging behaviour. The first track (AFS1) had 115 locations and presented a relatively consistent travel rate throughout its trip, interspersed with short foraging periods. The second track (AFS2) had 80 locations and travelled to and from a remote foraging location where it spent the majority of its time.

3.3 Results

3.3.1 Example Maps

A series of maps of location tracks and KDE derived UDs were created. These were used to demonstrate the effect of applying the unweighted and time-weighted kernels in combination with the three forms of smoothing parameter selection, h_{ref}, h_{LSCV} and h_{LHCV}. This series of KDE calculations was applied separately to two different Antarctic fur seal tracks, designated AFS1 (Figures **3.1** to **3.10**) and AFS2 (Figures **3.11** to **3.20**).

The tendency for the h_{ref} method to oversmooth a multi-modal distribution can be seen in Figures **3.1** and **3.11**. Figures **3.2** and **3.12** are produced using a time-weighted kernel which produces a more even distribution of the peaks in the KDE. This, when combined with the weighted version of the h_{ref} formula, h_{wref} appears to produce a more realistic result than the unweighted version.

The unweighted KDE with cross-validation smoothing by h_{LSCV} (Figures **3.3** and **3.13**) and h_{LHCV} (Figures **3.4** and **3.14**) produces a similar map for each of the smoothing parameter estimation methods. For both tracks, there was a slightly higher smoothing level for h_{LHCV}. The location density centric nature of the unweighted KDE is highlighted in the validation of complimentary location

subsets of AFS1 (Figures **3.5** and **3.6**). In both maps, the overlay of the two distributions indicates many inconsistencies between the distributions. However, the differences are slightly more for the h_{LSCV} method (Figure **3.5**) owing to its lower smoothing level. This effect is further accentuated in the validation maps for AFS2 (Figures **3.15** and **3.16**).

The time-weighted KDEs for h_{LSCV} across the two tracks, AFS1 (Figure **3.7**) and AFS2 (Figure **3.17**) shows a greater difference to their h_{LHCV} derived counterparts (Figures **3.8** and **3.18**) than was apparent in the unweighted versions. Also, the time weighted validation plots for AFS1 (Figures **3.9** and **3.10**) and AFS2 (Figures **3.19** and **3.20**) show greater overlap between the complementary distributions. This is particularly evident with the h_{LHCV} form of the KDE in both tracks (Figures **3.10** and **3.20**).

3.3.2 Validation Tests

Two way repeated measures analysis of variance were used to separately compare the validation test scores for BA and CV(RMSE) in response to kernel type (unweighted and time-weighted) and smoothing cross-validation type (h_{LSCV} and h_{LHCV}). All effects showed significant differences. For CV(RMSE), the results were kernel type ($F(99) = 178, p < .001$), smoothing cross-validation type ($F(99) = 758, p < .001$) and their interaction ($F(99) = 299, p < .001$). The results for BA were kernel type ($F(99) = 19, p < .001$), smoothing cross-validation type ($F(99) = 774, p < .001$) and their interaction ($F(99) = 53, p < .001$).

Paired samples t-tests were conducted separately for the difference and similarity scores to compare the results for the unweighted and time-weighted KDE validation tests. Additionally these results were shown in a series of line and parallel plots. The measure of difference between UD validation subsets for unweighted versus time-weighted KDEs shows that when using h_{LSCV}, CV(RMSE) only discriminates between these KDE methods when the unweighted UD comparisons have higher values (> 1.05) of CV(RMSE) (Figure **3.21**a,c) ($t(99) = 7.27, p < .001$). When this statistic was applied to AFS1 UDs generated with the h_{LHCV} smoothed KDE, the level of difference was consistently lower for the time-weighted KDE for each pairwise comparison to the unweighted KDE (Figure **3.21**b,d) ($t(99) = 17.01, p < .001$). This trend was less pronounced for

AFS2 and interestingly, lower values of CV(RMSE) for the unweighted KDEs show greater higher scores in the time-weighted measures (Figure **3.23**a,c) (t(99) = -281, p = .005). As with AFS1, the results for the difference test when applied to the h_{LHCV} KDE shows a more consistent lower CV(RMSE) value for the time-weighted KDEs (Figure **3.23**b,d) (t(99) = -8.42, p < .001). There was however, a small proportion that had higher time-weighted values.

The similarity score, BA, showed no consistent difference across the multiple test runs when comparing the validation of unweighted and time-weighted KDEs for AFS1 using h_{LSCV} (Figure **3.22**a,c) (t(99) = 0.67, p = .506). When applied to AFS2, the BA scores indicated a greater proportion of higher BA values for the unweighted KDE (Figure **3.24**a,c) (t(99) = 6.80, p < .001). The BA scores for the h_{LHCV} based KDEs showed a very consistent trend of increased similarity between the UD subset pairs for the time-weighted KDE. A similar result was found for AFS1 (Figure **3.22**b,d) (t(99) = -16.44, p < .001) and AFS2 (Figure **3.24**b,d) (t(99) = -12.48, p < .001).

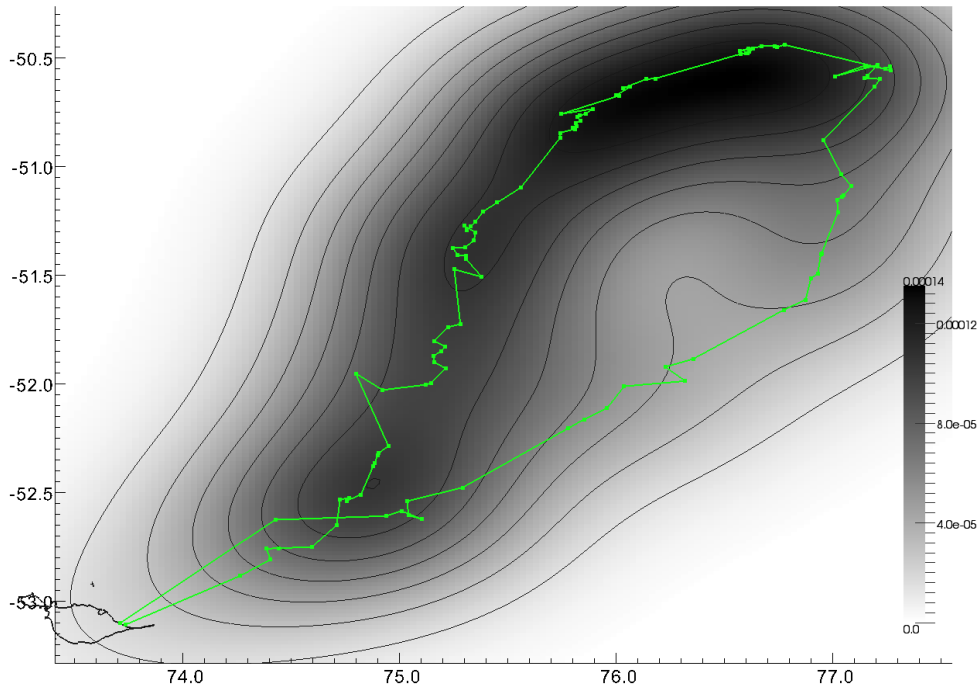

Figure 3.1. *Unweighted KDE of AFS1 locations using reference method, h_{ref}. Green points represent the Argos locations. Green connecting lines are included only for presentation purposes as interpolation was not incorporated into the calculation of the KDE. Grey scale shading indicates the normalised probability distribution with 10% contour lines overlaid. Heard Island is in the bottom, left corner.*

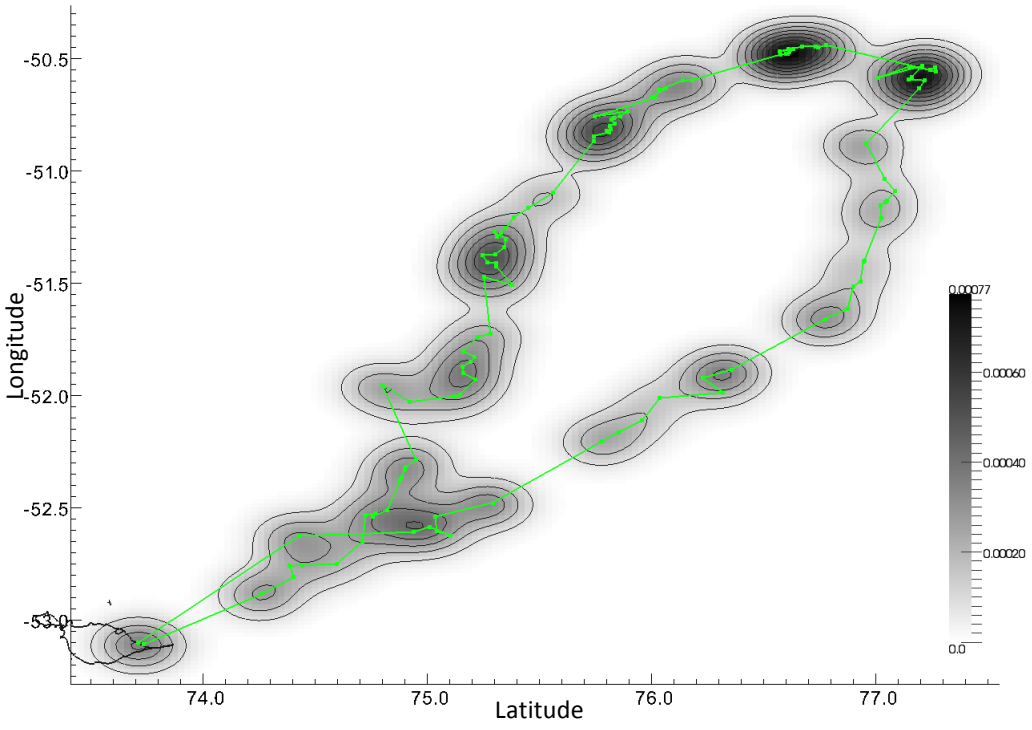

Figure 3.2. *Time weighted KDE of AFS1 locations using weighted reference method h_{wref}.*

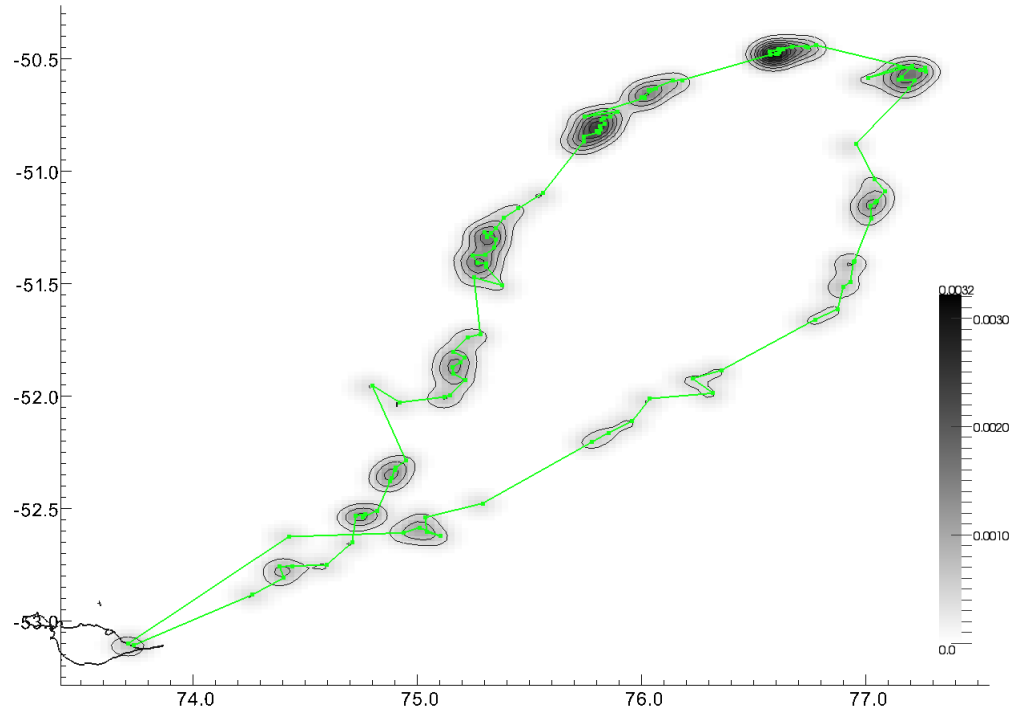

Figure 3.3. *Unweighted KDE of AFS1 locations. Smoothing determined by least squares cross-validation (h_{LSCV}).*

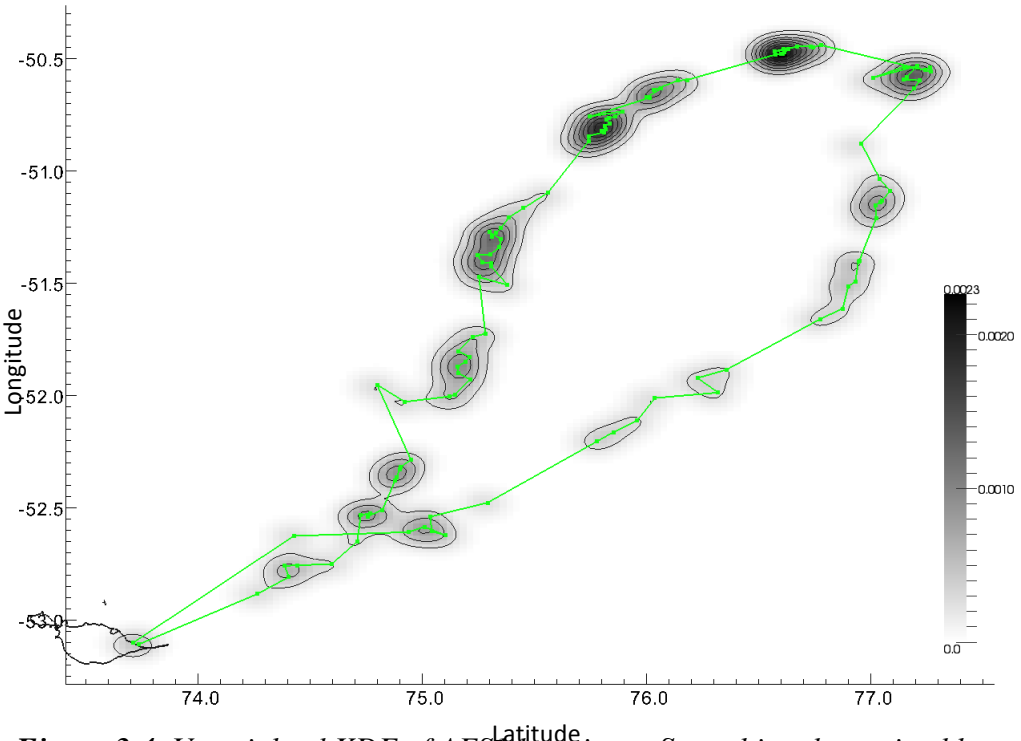

Figure 3.4. *Unweighted KDE of AFS1 locations. Smoothing determined by likelihood cross-validation (h_{LHCV}).*

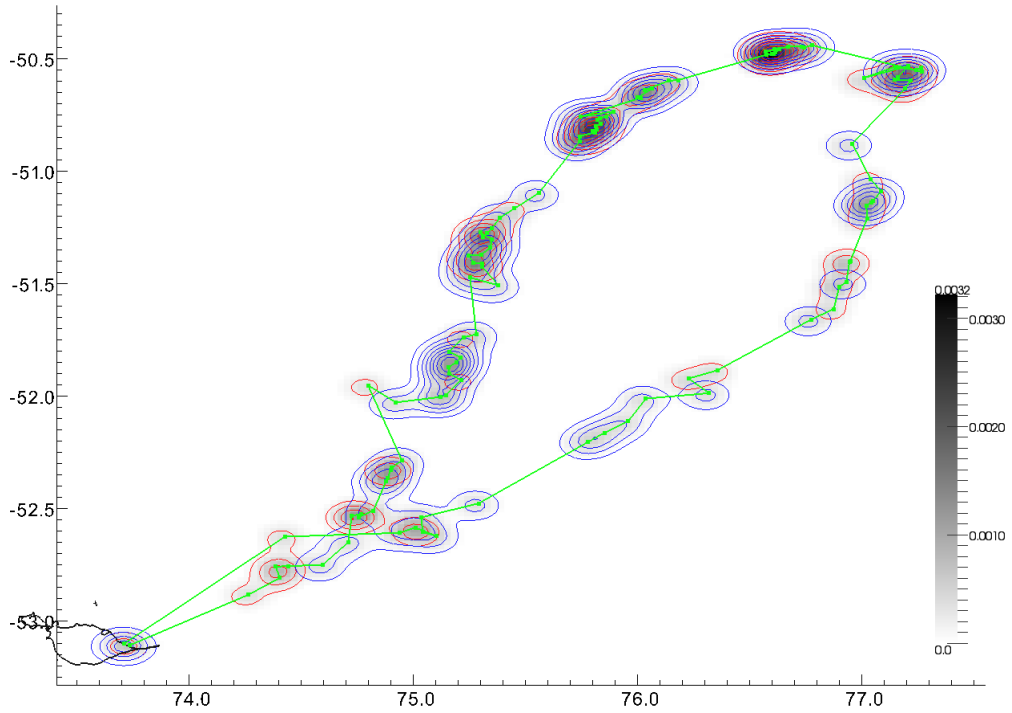

Figure 3.5. *Validation of UDs based on AFS1 subdivided locations. Red contour lines represent the UD of the first data subset while blue contours are used for the second data subset. Green points and lines define the track of the total set of locations and the grey scale is the UD for all locations. The UDs were constructed using unweighted KDEs and smoothing was determined with least squares cross-validation (h_{LSCV}).*

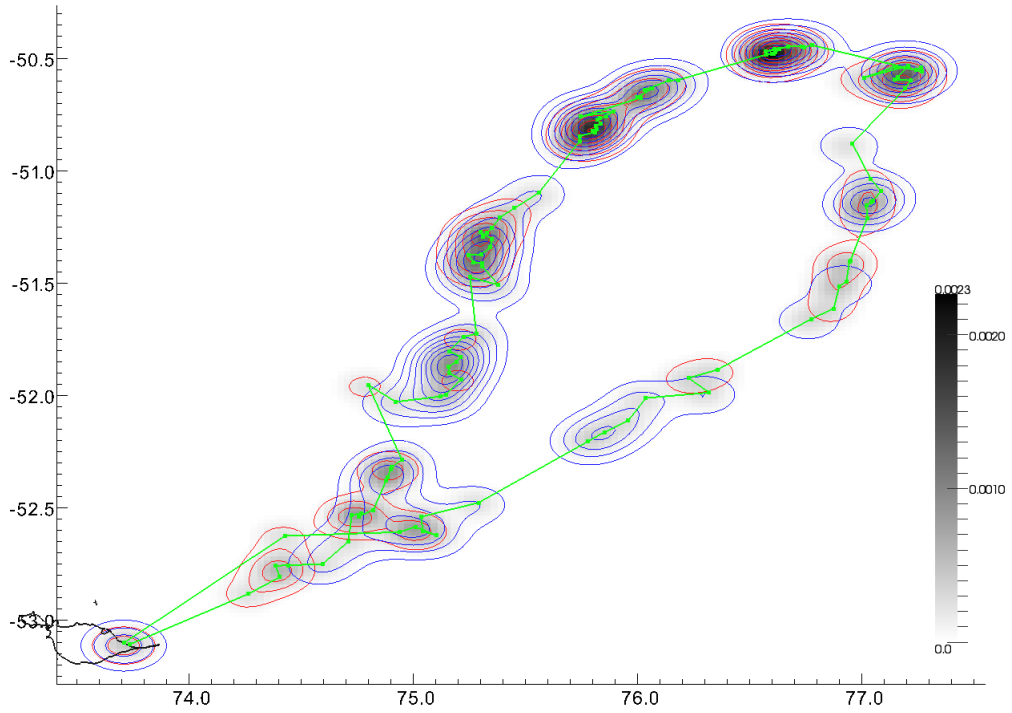

Figure 3.6. *UD validation of unweighted KDE with likelihood cross-validation (h_{LHCV}) using AFS1 data.*

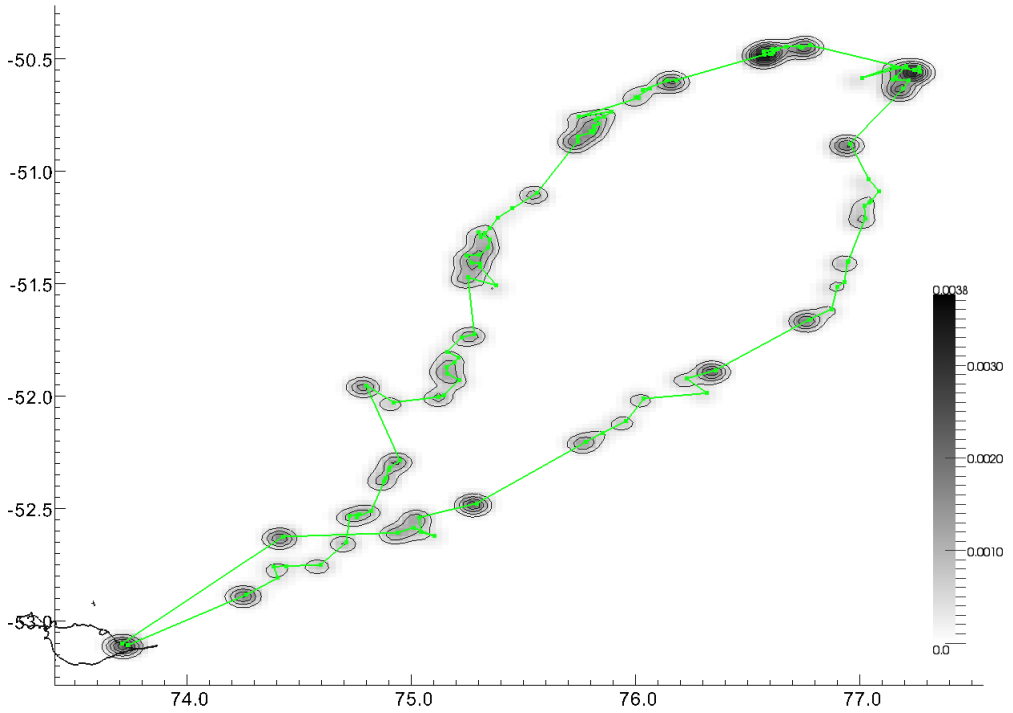

Figure 3.7. *Time-weighted KDE of AFS1 locations. Smoothing determined by least squares cross-validation (h_{LSCV}).*

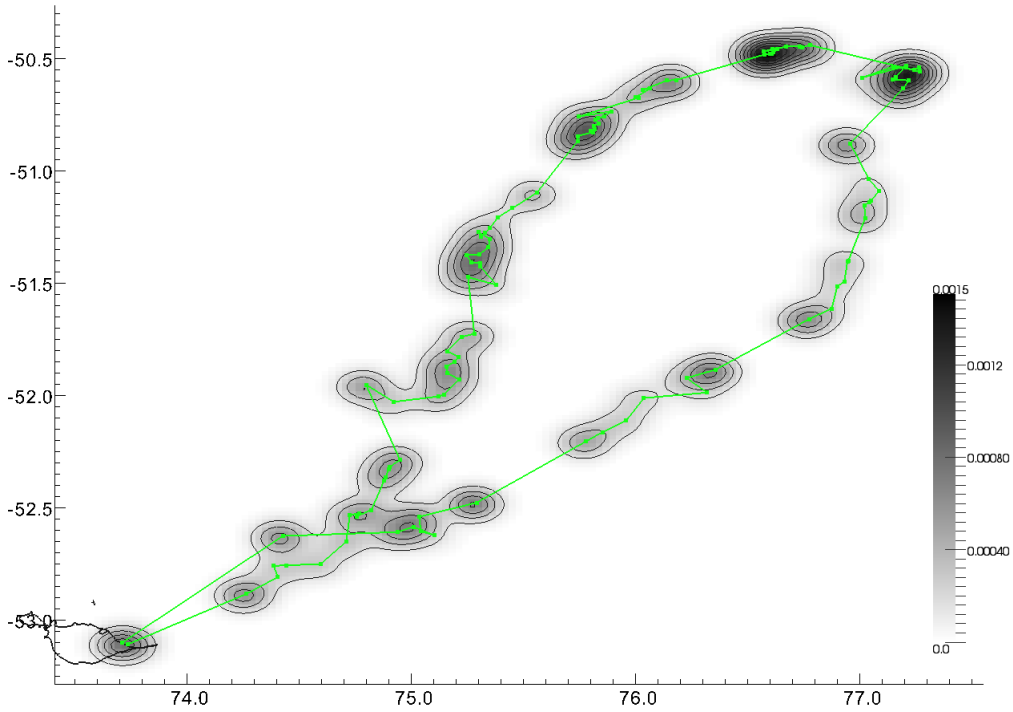

Figure 3.8. *Time-weighted KDE of AFS1 locations. Smoothing determined by likelihood cross-validation (h_{LHCV}).*

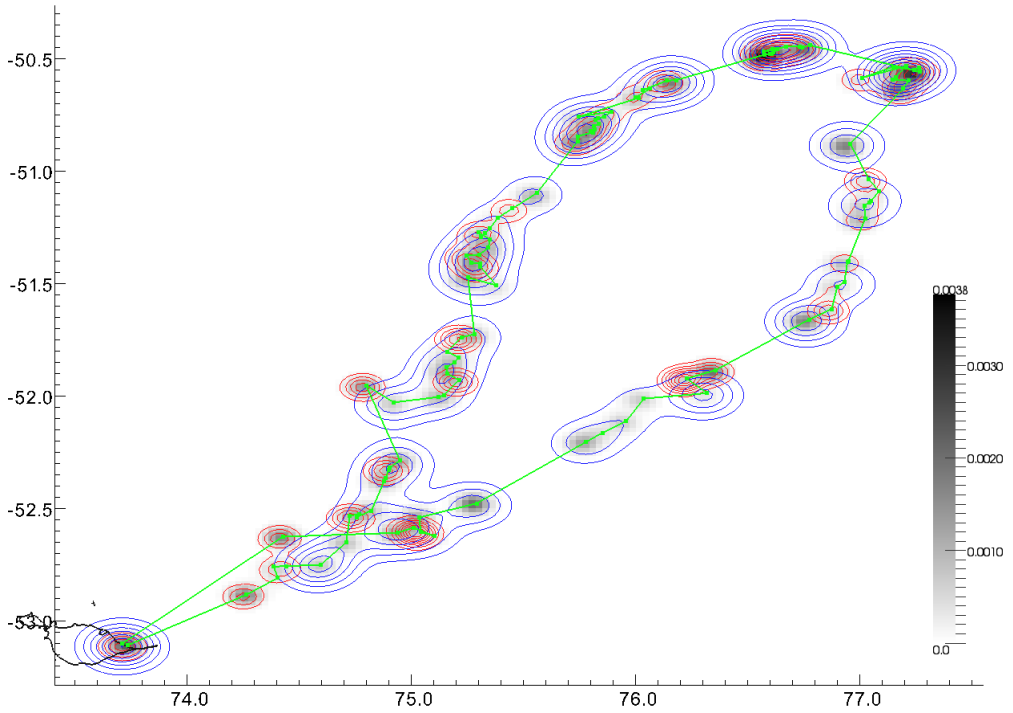

Figure 3.9. *UD validation of time-weighted KDE with least squares cross-validation (h_{LSCV}) using AFS1 data.*

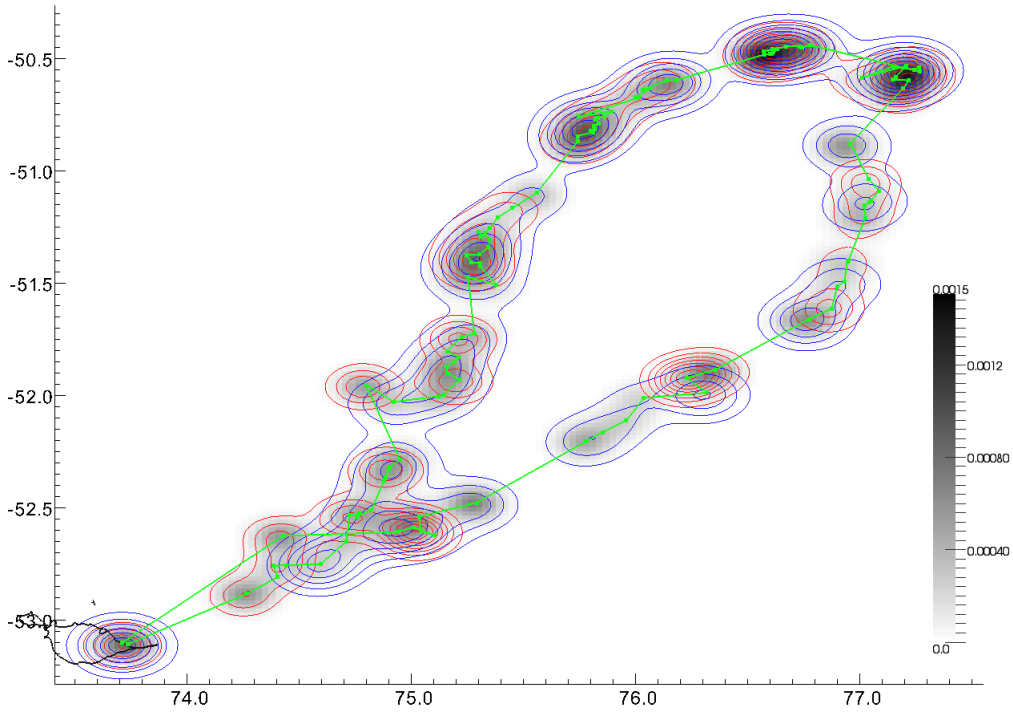

Figure 3.10. *UD validation of time-weighted KDE with likelihood cross-validation (h_{LHCV}) using AFS1 data.*

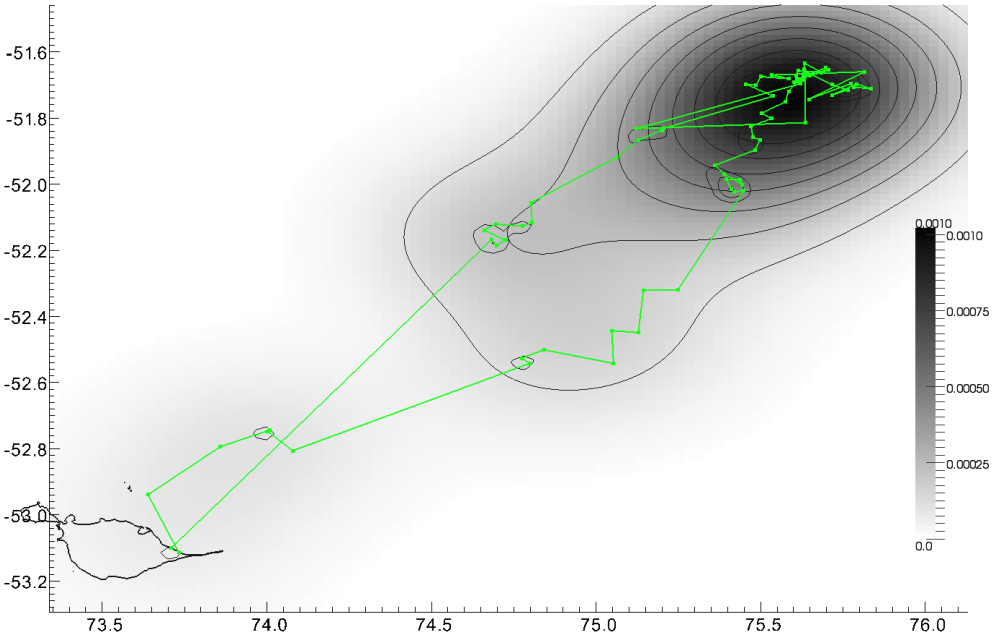

Figure 3.11. *Unweighted KDE of AFS2 locations using reference method,* h_{ref}.

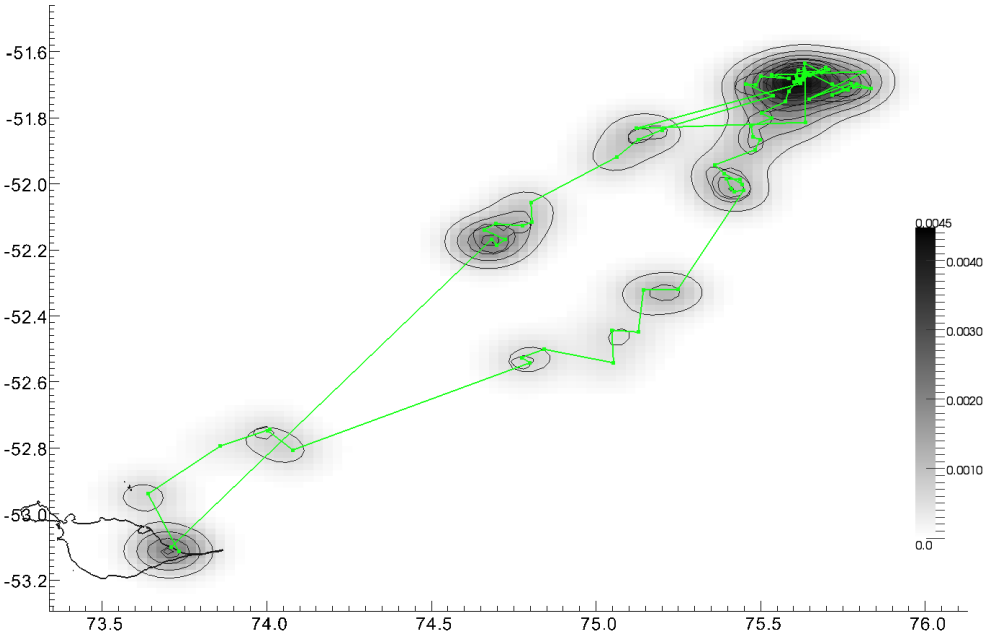

Figure 3.12. *Time weighted KDE of AFS2 locations using weighted reference method* h_{wref}.

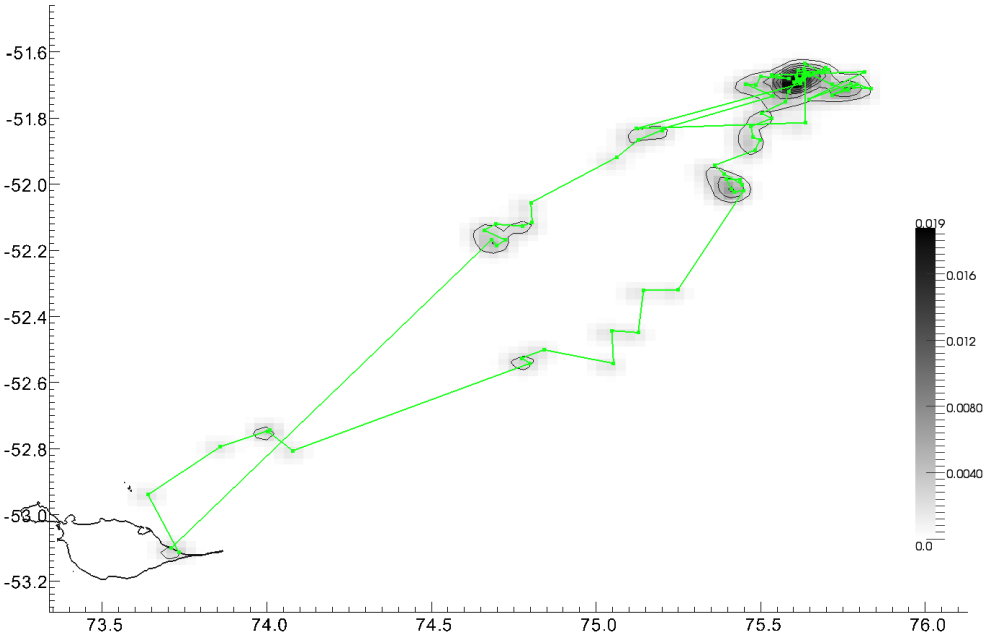

Figure 3.13. *Unweighted KDE of AFS2 locations. Smoothing determined by least squares cross-validation (h_{LSCV}).*

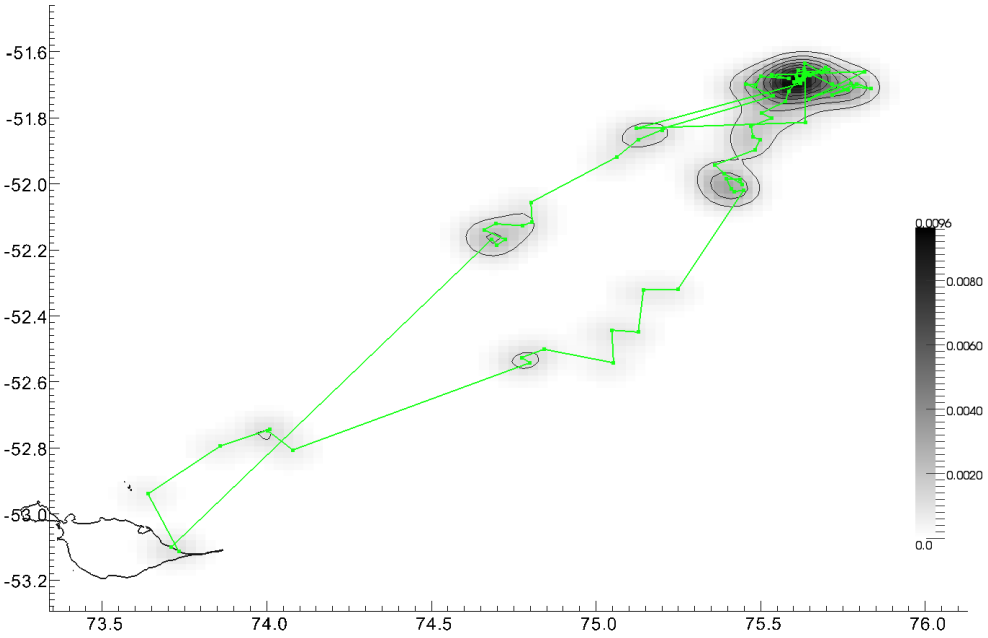

Figure 3.14. *Unweighted KDE of AFS2 locations. Smoothing determined by likelihood cross-validation (h_{LHCV}).*

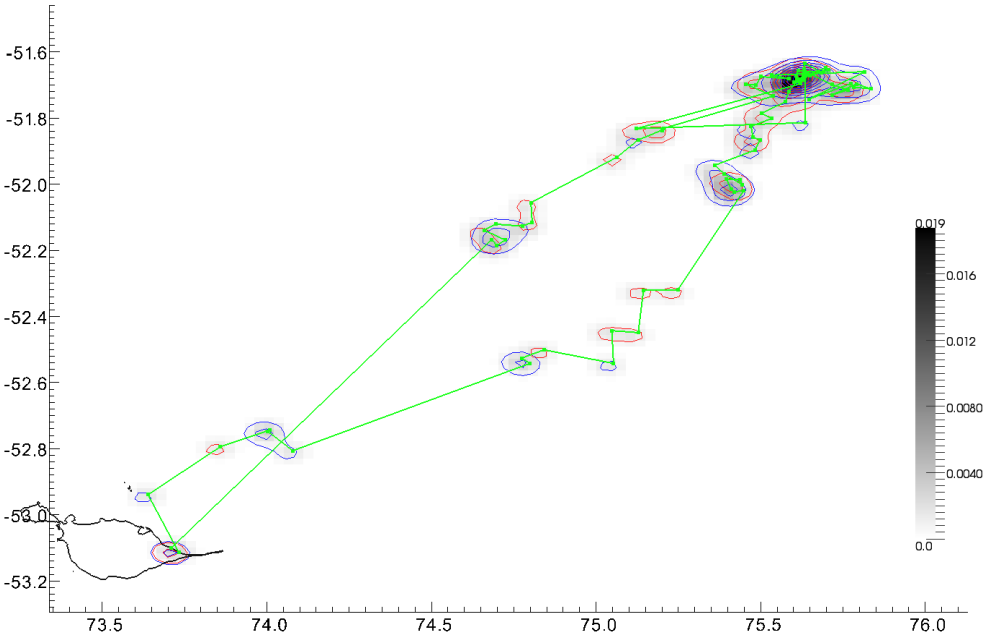

Figure 3.15. *UD validation of unweighted KDE with least squares cross-validation (h_{LSCV}) using AFS2 data.*

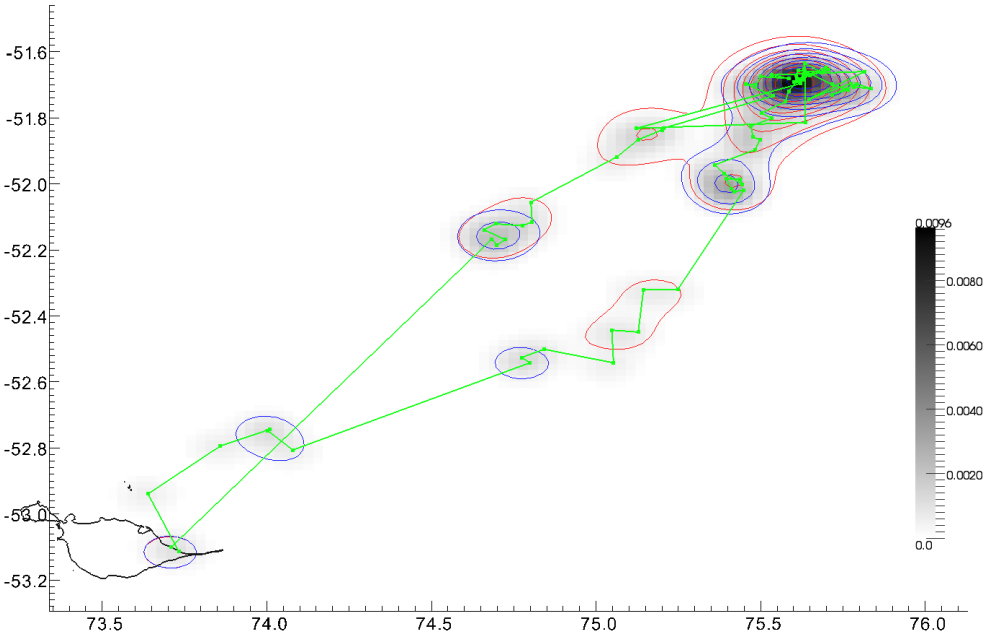

Figure 3.16. *UD validation of unweighted KDE with likelihood cross-validation (h_{LHCV}) using AFS2 data.*

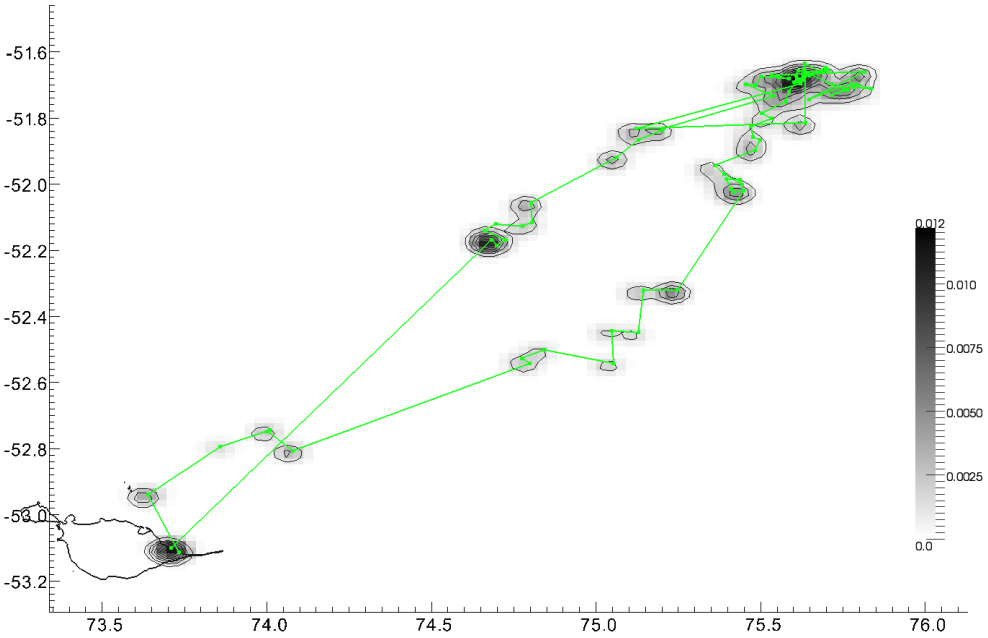

Figure 3.17. *Time-weighted KDE of AFS2 locations. Smoothing determined by least squares cross-validation (h_{LSCV}).*

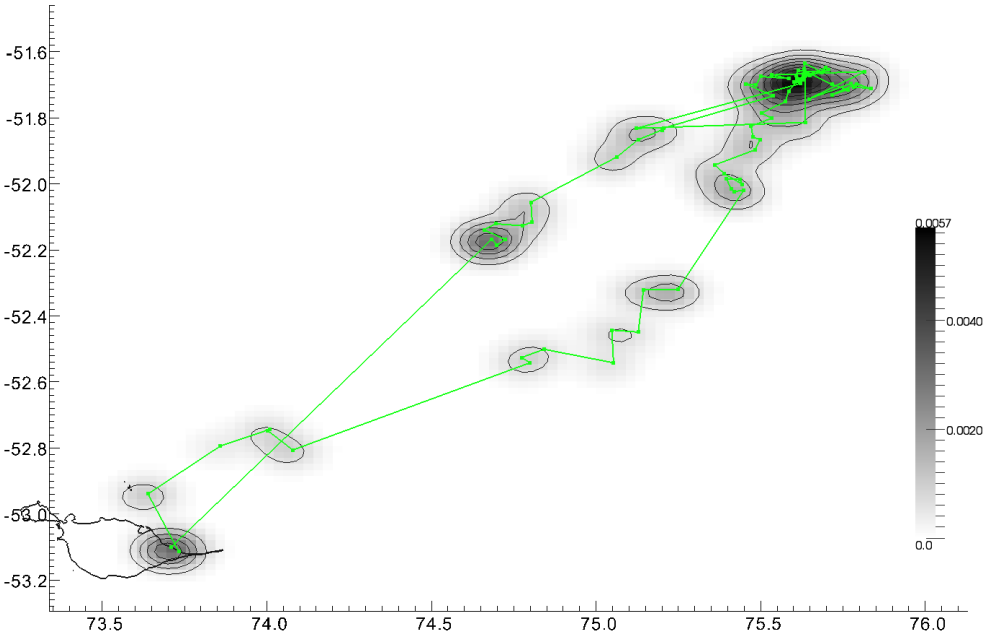

Figure 3.18. *Time-weighted KDE of AFS1 locations. Smoothing determined by likelihood cross-validation (h_{LHCV}).*

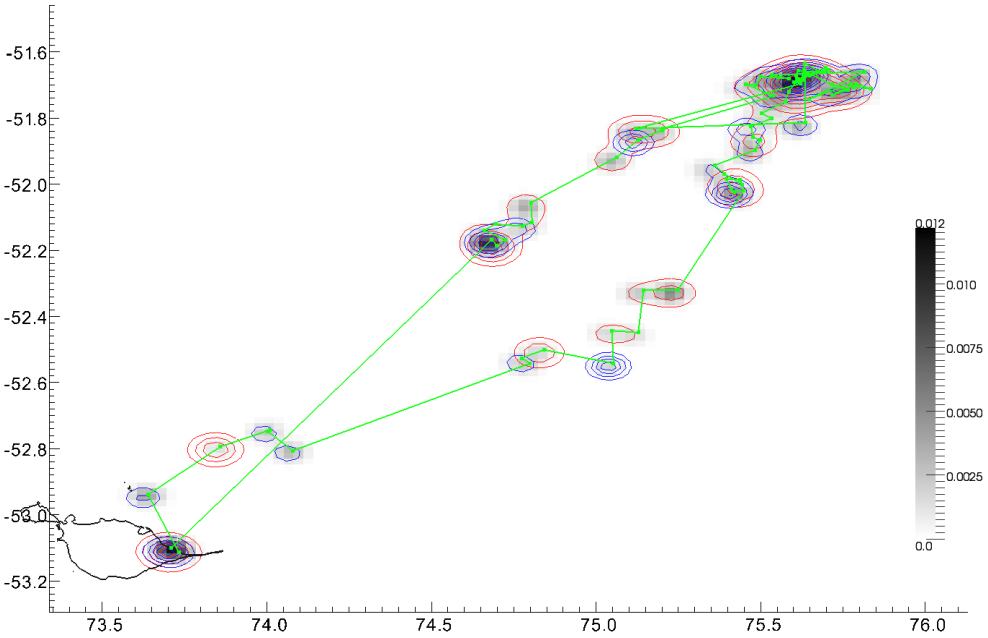

Figure 3.19. *UD validation of time-weighted KDE with least squares cross-validation (h_{LSCV}) using AFS2 data.*

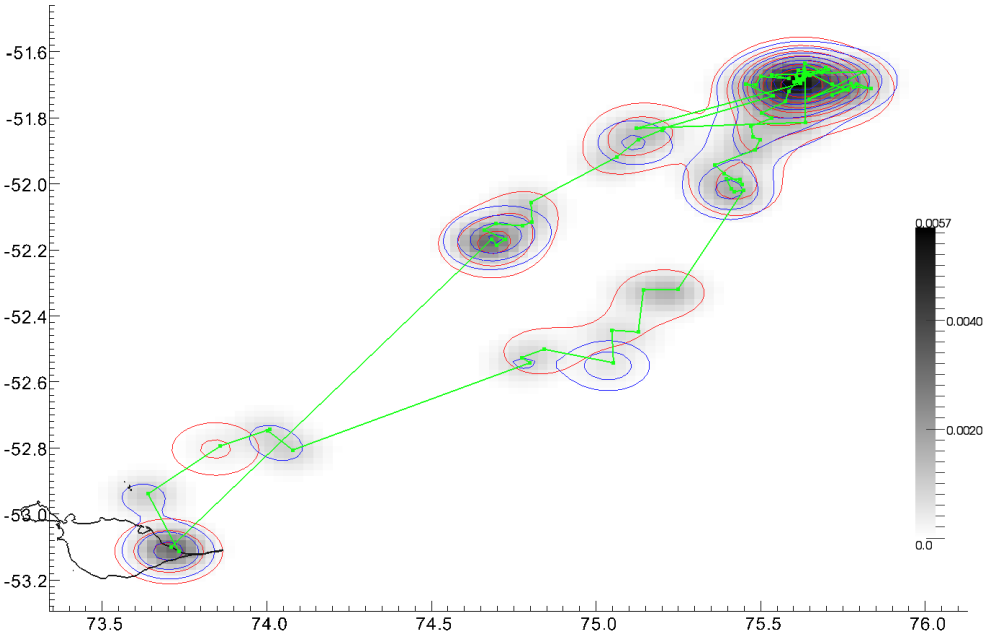

Figure 3.20. *UD validation of time-weighted KDE with likelihood cross-validation (h_{LHCV}) using AFS2 data.*

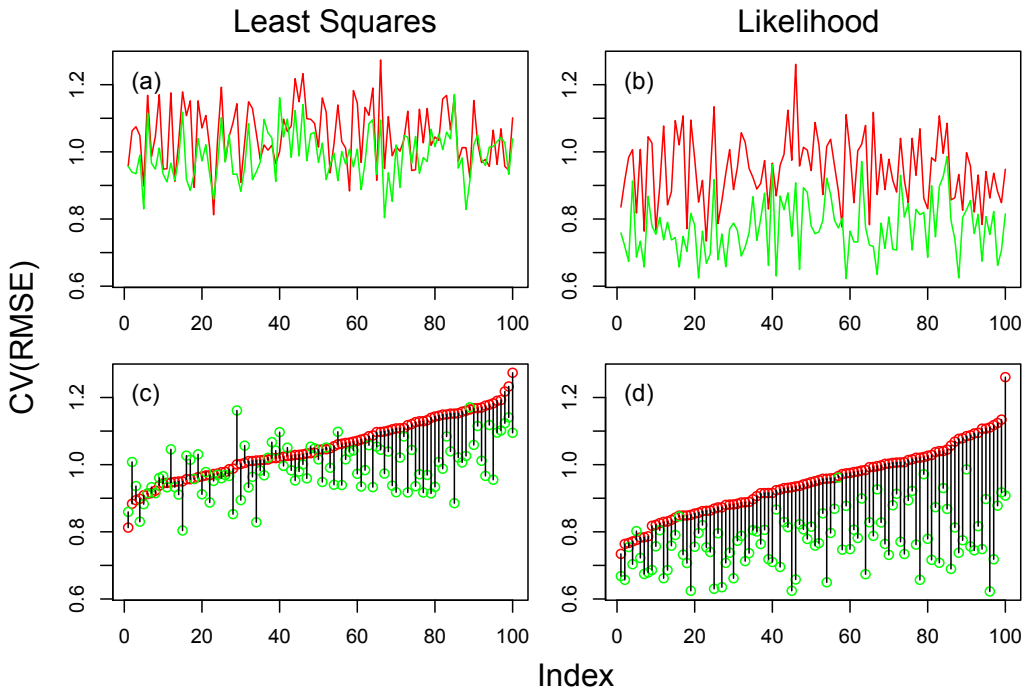

Figure 3.21. *Validation tests of UD subsets repeated 100 times for track, AFS1. Difference between UDs measured by CV(RMSE). Sections (a) and (b) show the raw output values. Sections (c) and (d) represent the paired nature of this data as parallel plots. Least squares cross-validation is shown in sections (a) and (c) while likelihood cross-validation is in sections (b) and (d). Red lines and circles represent unweighted kernel method and green is for the time-weighted kernel.*

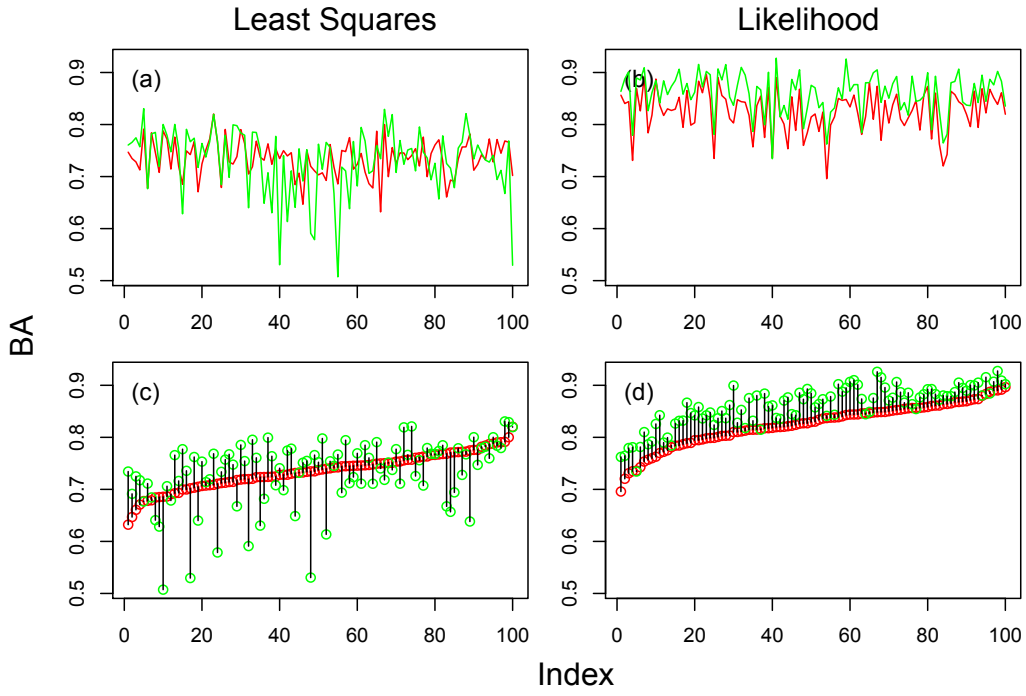

Figure 3.22. *Validation tests of UD subsets for track, AFS1. Similarity measured by BA.*

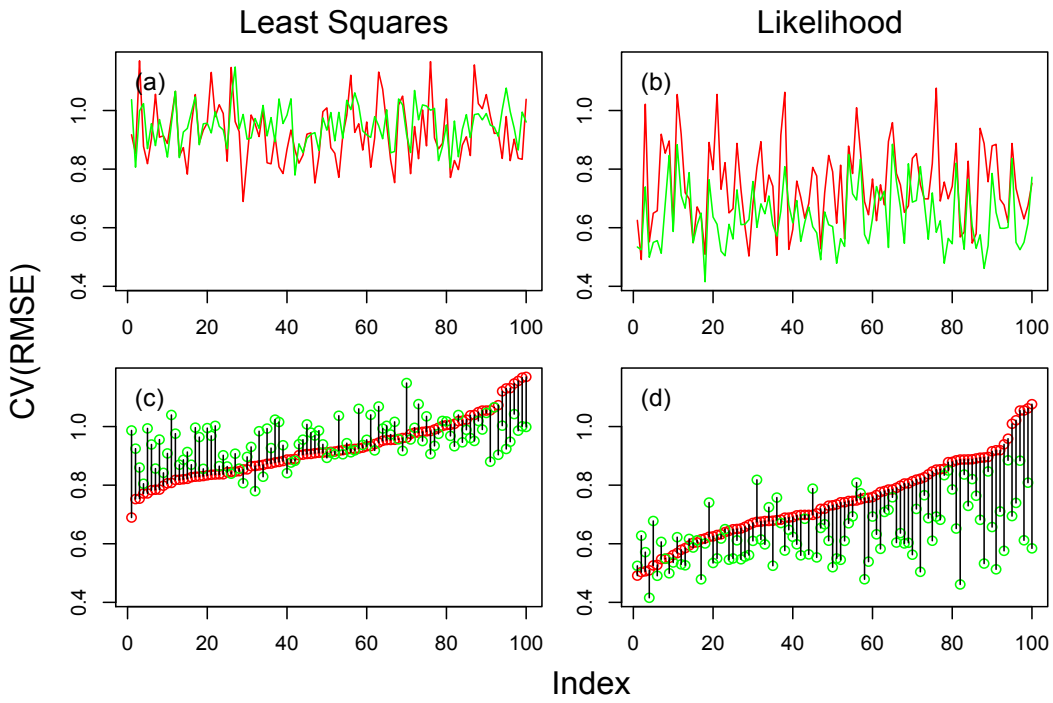

Figure 3.23. *Validation tests of UD subsets for track, AFS2. Difference measured by CV(RMSE).*

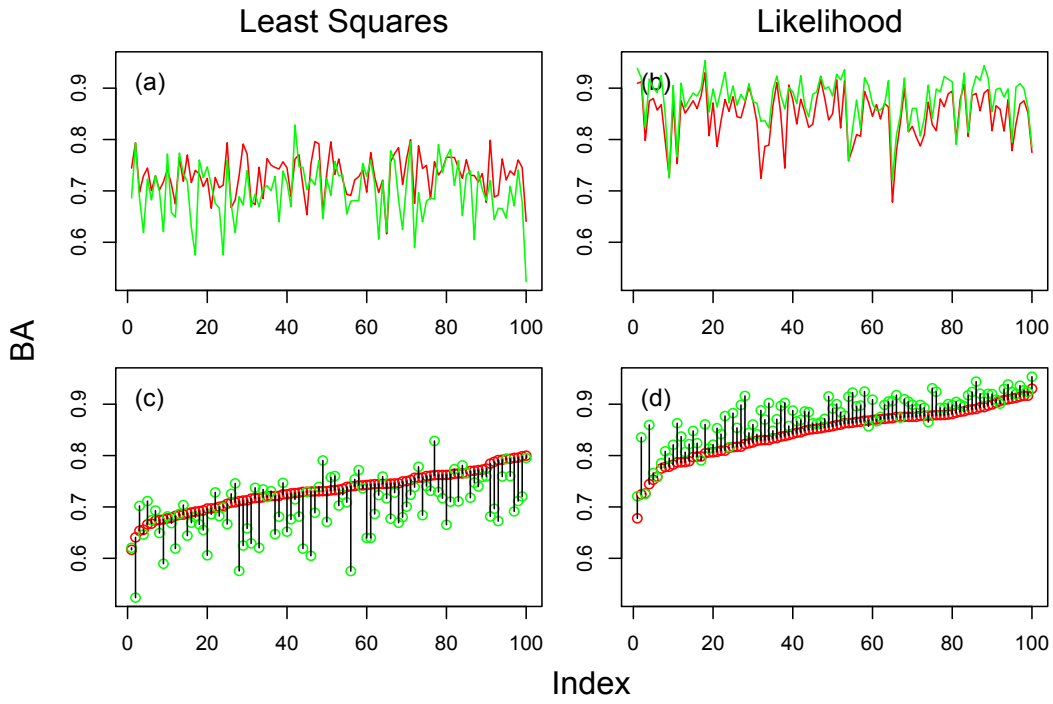

Figure 3.24. *Validation tests of UD subsets for track, AFS2. Similarity measured by BA.*

3.4 Discussion

Estimation of an animal's UD from sparse and irregular location data requires the application of a number of analytical steps. At any stage in the analysis, more than one option can exist and an incorrect decision may lead to false or misleading results. It is therefore vital that a researcher applies a methodology with knowledge of the consequences of their decisions and with confidence that the statistic's behaviour has been quantitatively defined.

Any opportunity to simplify analytical techniques can reduce a researchers work load as well as providing less opportunity for the introduction of error. Since animal tracking in the marine domain often involves the use of the Argos system, a useful addition to the kernel smoothing literature is the modification of the underlying mathematics so as to allow for the direct use of geographical coordinates. While not strictly necessary, the absence of Cartesian data transformations not only simplifies the use of kernel smoothing for Argos data but may have the potential to improve accuracy since the curvature of the Earth is implicitly accounted for via the use of great circle distances.

An important consideration in the use of kernel smoothing is the amount of smoothing which is controlled by setting a smoothing parameter (h). While setting this value manually may be adequate for qualitative or broad scale analysis (e.g. Chapter 2), automatic data driven methods should also be considered. Three techniques in common use are the reference method (h_{ref}), least squares cross-validation (h_{LSCV}) and likelihood cross-validation (h_{LHCV}). Although the use of h_{ref} in location data analysis has been discounted owing to its tendency to over smooth multi-modal data (Worton 1995), it is necessarily described in this chapter as it is the first step in determining the other two methods, h_{LSCV} and h_{LHCV}.

The use of unweighted location data in the creation of a kernel density estimate implies that the influence of each data point is equal to every other data point. Where location data is taken to represent a portion of a temporal sequence of movement and where that data is infrequently and sporadically obtained, the assumption of equal influence would seem to be false. Since Argos data is heavily influenced by the animal's behaviour, e.g. diving, resting, travelling as well as the weather and the technical limitations of the system, the timing of

acquisition of location data is inherently irregular. Techniques exist that compensate for this by either discarding data that does not fall within a regular time window (Edrén et al. 2010) or that reduce multiple locations to a regular summary statistic such as a daily mean (Seney & Landry 2008; Hart et al. 2010) or median (James et al. 2005). Since these methods result in the loss of information from an already sparse data set, the use of time weighting both here and in other studies (Katajisto & Moilanen 2006) is being promoted since it provides the opportunity for each data point to be retained and therefore to make a contribution to the overall result.

A KDE derived UD's ability to predict unknown locations was assessed using 100 repeated validation tests (Section 3.3.2) of the randomly subdivided animal tracks, AFS1 and AFS2. The effectiveness of various smoothing methods was determined with the two statistics, CV(RMSE) to measure difference and BA for similarity. When applied to the exact same data, these statistics usually though not always provided opposite results. This indicates that they are not just the inverse of each other but detect different properties in the data. It follows that the KDE technique that shows the most consistent results with the lowest CV(RMSE) and highest BA across most or all of the 100 tests should be the best estimator of unknown locations. When tested across all combinations of unweighted and time-weighted kernels against h_{LSCV} and h_{LHCV} smoothing methods, the results strongly suggest that the best approach to generating a UD from KDE is via an appropriately selected weighting of data points in conjunction with h_{LHCV} smoothing in preference to h_{LSCV}. This finding was consistent across both tracks, AFS1 and AFS2. Furthermore, the use of h_{LHCV} smoothing in preference to h_{LSCV} is supported by recent findings in the literature (Matthiopoulos 2003; Horne & Garton 2006a).

This chapter has provided a methodology for the direct inclusion of geographical data into kernel smoothing algorithms that till now have required preliminary conversion to Cartesian coordinates. In addition to defining the equations for weighted and unweighted kernel types, the three main forms of smoothing parameter estimation have been adapted to both weighted and geographical data. Exploration of the behaviour of these equations when applied to two representative fur seal tracks found that the most effective and consistent

form of UD was constructed from a weighted kernel using likelihood cross-validation to determine the level of smoothing. In the next chapter, I will explore the addition of interpolation using both linear and model driven approaches and compare the resulting UDs to those produced here.

Chapter 4

The Use of Model Interpolated Kernel Smoothing in Animal Tracking Studies

Abstract

Quantifying an animal's utilisation of space from a small sample of locations is confounded by a number of issues. The nature of these data are such that they are temporally and spatially sampled on an infrequent and irregular basis as well as being distorted by a potentially large error component. By using interpolation, intermediate locations can be estimated and a more evenly distributed data set can be derived. Traditionally, two-dimensional probability distributions, or utilisation distributions (UD) are generated by applying kernel smoothing techniques directly to location data. In this chapter I present a new technique for interpolating location data prior to smoothing. This reduces the influence of individual Argos locations on the resulting UD by introducing interpolated pseudo-data. This method uses a multi-objective evolutionary computation algorithm called extremal optimisation as a mechanism for searching the potential paths that an animal could move through. The algorithm incorporates empirically determined estimates of location error and combines these with a population-derived probability distribution function of animal movement speeds. This information is used to control the distribution of interpolated pseudo-locations as a pre-cursor to creating a UD through the application of a weighted kernel density estimate. Validation tests were performed by randomly dividing a track into two sub-tracks and using one to predict the other. When tested for its ability to predict a UD generated from an unknown set of locations, this technique known as model interpolated kernel smoothing was found to out-perform unweighted, time-weighted and linearly interpolated UDs.

4.1 Introduction

The tracking of animal movement through the use of telemetry tags provides researchers with a valuable tool for understanding an animal's utilisation of space. A common approach to obtaining these data is through the attachment of radio transmitters in conjunction with the Argos satellite system (e.g. Goulet et al. 1999; Guinet et al. 2001; Le Boeuf et al. 2000). The Argos system, which estimates location through Doppler shift measurements of transmitter carrier signals, is affected by a number of factors including the number and position of satellites, the weather and the animal's behaviour such as diving, resting or travelling. Consequently, an inherent limitation in this form of tracking is the level of error associated with location fixes and the sporadic and infrequent acquisition times of these locations (Argos 2008; Freitas et al. 2008).

Argos locations are usually supplied with an estimate of their potential error known as a location class. The size of the error are estimated to range from < 250m through to an unknown distance and defines the radius of a circle that encompasses the first standard deviation of error estimates (Argos 2008). These errors are primarily a function of the number of radio transmissions received by the orbiting satellites. The more messages received, the more accurate the location class. Field based experiments designed to empirically measure the error rate have refined the location class error estimates to an elliptical shape which is longitudinally elongated and whose size varies in accordance with the class of the location (Vincent et al. 2002). This error is problematic to any analysis that depends on the location's position in the production of an estimated movement path. It is therefore important to consider techniques that quantitatively assess and incorporate this error and its effect on the calculated output. CLS Argos have recently implemented the use of a new algorithm to estimate transmitter location (Argos 2008), but issues of spatial error remain. In this chapter I deal with data obtained during 2003/04 that used the older system of location estimation and location class errors. However, the analysis detailed in this chapter will function equally well with the new Argos algorithm.

Reconstruction of the path that an animal travelled along when based on a sample of locations, can be achieved using interpolation (Lonergan et al. 2009). Interpolation of a path predicts the intermediate locations and therefore can be

used to address the issue of uneven sampling (Tremblay et al. 2006). The most commonly used method for interpolating locations is linear interpolation. By connecting each location pair with a straight line, this technique imposes an assumption of straight line travel at a constant speed. Although simple to implement and intuitive to visualise, this approach imposes a great deal of influence on each location as well as making a strong quantitative statement about the certainty of both the Argos location and the intermediate positions. While this may be acceptable and even appropriate at small spatial and temporal sampling scales, the larger the gap between locations or the greater the temporal irregularity, the less likely it is that the animal moved in a straight line (Turchin 1998). More sophisticated approaches to location interpolation such as curvilinear interpolation (Tremblay et al. 2006) and state space models (Jonsen et al. 2003) have also been proposed. When applied to location track data, these techniques have been used to provide a more accurate intermediate location estimate than linear interpolation.

By considering a set of locations as a sample of autocorrelated locations along an individual's path of travel, it can be seen that a large component of the total variance in these data are driven by behaviour (Turchin 1998). Where multiple tracks of similarly grouped individuals, e.g. same sex seals from the one colony, are combined, it becomes possible to infer population level measures of behaviour. These measures can in turn be used to guide the estimation of movement in individual paths (Jonsen et al. 2003). Another issue in predicting the actual path is the level of error associated with each location fix. By integrating models of behaviour with models of observation error, the potential exists for the creation of a more accurate, data driven and defendable estimate of the path of movement. One approach to achieving this is through the use of state space models that predict the future state of an animal based on its current state (Patterson et al. 2008). An alternative method presented here is to use an evolutionary computation based algorithm for performing large scale multi-objective function searches. This system interprets speed as a simple scalar measure of behaviour. It then uses a population derived probability distribution function (PDF) of inter-location travel speeds to guide the search algorithm and produce an interpolated set of pseudo-locations (PL). Such a system can also be

used to incorporate the Argos location estimation error into its model of movement.

A probability map that represents an animal's spatial use is known as a utilisation distribution (UD) (Matthiopoulos 2003). The usual method for generating a UD from animal location data is to smooth the data by applying kernel density estimation (KDE) to the location data set (Worton 1989, 1995). The performance of KDE is strongly influenced by the quality and quantity of the data it is built from as well as the method used to determine the level of smoothing. In Chapter 3, I presented various applications of KDE for the generation of UD maps and demonstrated how significant improvements in performance can be achieved when informed design decisions are made. In particular, I showed the need for accurately representing the relative influence of locations through the process of kernel weighting. I also found that the ability to predict unknown locations was significantly improved when the smoothing parameter was determined with likelihood cross-validation rather than the more commonly used technique of least squares cross-validation. In this chapter, I describe a method for interpolating an animal's movements based on a model of population level estimates of behaviour and empirically derived estimates of location error. This chapter extends the methods and findings of Chapter 3 by using KDE to generate a UD based on the output of the interpolating model. Matthiopoulos (2003) described a technique called model supervised kernel smoothing which allowed for the quantitative inclusion of additional information into the structure of a standard kernel smoothed distribution. In a similar vein, the technique in this chapter also provides for the incorporation of additional information to a spatial UD and is therefore entitled model interpolated kernel smoothing (MIKS).

4.2 Methods

4.2.1 Data Collection

The location data used in this study were obtained from the Antarctic fur seal (*Arctocephalus gazella*) colony at Spit Bay, Heard Island (53° 07′ S, 73° 44′ E) between December 2003 and February 2004. All data was received using the Argos satellite system. Platform terminal transmitters (PTT) manufactured by

Sirtrack Limited were attached to individual female fur seals. The PTT was glued to the back of the animal using Araldite 2017 epoxy adhesive (Vantico, Switzerland). The PTT was set to transmit on a continuous duty cycle with a 30 second repetition rate (Robinson et al. In Prep).

4.2.2 Speed Filter

Pre-processing of the data to remove unfeasible high speed locations was achieved using the filter technique described in McConnell et al. (1992). All data was filtered at 3.0 m/s as determined by Bonadonna et al. (2000). By applying the formula,

$$V_i = \sqrt{\frac{1}{4} \times \sum_{j=-2, j\neq 0}^{j=2} \left(v_{i,j+i}\right)^2}, \tag{4.1}$$

velocity between adjacent locations is calculated. Once the velocity of all locations in a track is calculated, the location with the highest speed is removed if it exceeds the maximum speed. This is repeated until all velocities are less than the maximum speed.

Using the same method as Chapter 3, time spent at the haul out site was excluded from the track. This was achieved by removing all locations that fall within a circle of radius 5 km that is centred at the position of the haul out site. In order to preserve the travel to and from this site, the last location within the circle before the track begins and the first location within the circle after returning were not excluded.

4.2.3 Optimisation Interpolation

Mathematical optimisation of a function usually refers to the search for a maxima or minima. Broadly speaking, optimisation algorithms can be classed as exact or stochastic. Exact techniques require knowledge of the underlying mathematical function (e.g. integral calculus) or involve a thorough and complete, iterative exploration of the search space. Such a search is computationally intensive and therefore restricted to relatively small domains. As a result, exact techniques are not usually applicable to high dimensional, multi-objective systems. A faster and more flexible approach is to incorporate a random element into a guided search of the domain. These stochastic techniques are fast and

computationally efficient. They provide near optimal solutions to problems that would otherwise be considered intractable if attempting to solve exactly (Chiong 2009).

Extremal optimisation (EO) is a relatively new stochastic search and optimisation algorithm that is based on the Bak-Sneppen model of species co-evolution (Paczuski et al. 1996; Boettcher & Percus 2000). In this model, competition amongst interacting species causes a general trend of overall increasing fitness by the repeated mutation of the least fit species. This algorithm, like all evolutionary optimisation algorithms, treats a function's set of parameters as variable 'genetic' components whose 'phenotype' is generated by one or more assessment equations known as objective or fitness functions. An iterative process of random variation (usually referred to as mutation) is followed by some form of fitness based selection. By seeking to minimise (or maximise) the objective functions, a near optimal solution can be generated. The advantage of these combinatorial optimisation algorithms is that problems that are otherwise considered computationally intractable due to the sheer volume of calculations required can be resolved in a relatively short period of time. The trade-off is that the solution is almost always closely approaching but not exactly the definitive optimal solution (Osyczka 2002). EO differs from many of the more routinely used evolutionary algorithms such as genetic algorithms (GA) (Mitchell 1998), particle swarm optimisation (PSO) (Moore & Chapman 1999) and ant colony optimisation (Marco & Gianni 1999) by only working on a single solution rather than a population of interacting solutions and by making only a single parameter change per iteration. Also, as in the Bak-Sneppen model, it depends on mutation of the least adapted species rather than the more common practice of selecting for the best adapted.

At any given instant, the state of the EO algorithm represents a single candidate solution. This solution state is defined by a vector, $x = (x_1, \ldots, x_P)$ of P parameters. The purpose of the EO is to find a set of parameters, x that minimise a fitness function, λ. EO works by making a single change, known as a mutation, to a parameter, x_i and then assessing the effect of this change using λ. This mutation is then reversed, thereby returning the system to its current state. This is repeated for all components of x and then one of the mutated components is

selected. The selected component's mutation is then incorporated into the vector, x and the next iteration begins.

A critical part of the success of this approach is the method employed for selecting the mutated component, x_i. A property of both the Bak-Sneppen model and EO is the concept of self organised criticality (Bak et al. 1988). This is a form of emergent behaviour that can occur in systems built from components with a high degree of local interaction. It results in occasional and unpredictable mass shifts in fitness as the system collapses to a new stable state in much the same way as a snow avalanche forms (Anderson et al. 2004). The typical extremal optimisation implementation simulates this effect as a way of escaping local minima. This is achieved by ranking and ordering all mutations according to their relative fitness scores and randomly selecting a single mutation with a probability that is determined by a power law distribution (Boettcher & Percus 2004). At each iteration of the system, a rank ordered mutation has a probability, P of being selected according to

$$P(i) \sim i^{-\tau} \quad (1 \leq i \leq n;\ \tau > 0) \tag{4.2}$$

where i is the mutation's position in the rank order and n is the number of mutations. Adjustment of the constant, τ controls the search behaviour of the EO algorithm. A value of 0 would produce equal probability for all selections, thereby negating the selection criteria and the influence of the fitness function. This would produce a system that is essentially random and with no ability to optimise. As the value increases, the selection becomes gradually more deterministic. By approximately, $\tau = 6$, only $i = 1$ is selected. In this case, the EO would rapidly converge on the nearest solution which depending on the nature of the problem space could easily result in a non-global solution. In practice, this value is usually determined experimentally and is tailored to the problem at hand.

Optimisation of a system with a single objective is relatively straight-forward in so far as the assessment of the quality of one solution over another is achievable through a simple arithmetic comparison such as $x < y$. Extension of an optimisation algorithm into the multi-objective paradigm is complicated by the need to satisfy the requirements of more than one objective that may have conflicting values. This can be illustrated by a simple case of finding the

minimum value for two functions, $f(x) = x$ and $g(x) = x^2$ where the domain is $-1 \leq x \leq 1$ (see Figure **4.1**). Assume the optimisation starts at $x = 1$. As x moves from 1 to 0, both functions return continuously decreasing values, thereby signalling that the changes to x are correct. However, once the value of x falls below 0, $f(x)$ continues to decrease, while $g(x)$ starts to increase as it is now moving away from its minimum. In other words, improving the solution for one objective cannot be achieved without reducing the solution for another. Clearly there are two equally correct solutions, $x = 0\ for\ g(x) = 0$ and $x = -1\ for\ f(x) = -1$. In a real world system, this is often further complicated by non-linear functions that may contain multiple local and global minima (or maxima). In such a system, there is no single solution but rather a set of equally viable solutions. These solutions are said to be Pareto optimal and the set of Pareto optimal solutions is termed the Pareto front (Fieldsend & Singh 2002).

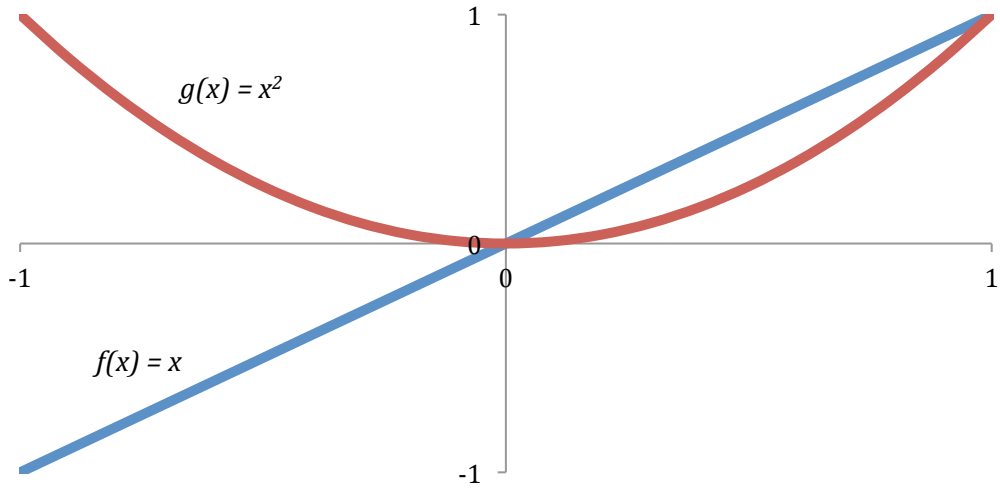

Figure 4.1. *Example of a simple multi-objective problem.*

Given a general multi-objective optimisation problem containing D objectives: $\lambda_d(x), d = 1, \ldots, D$, the goal is to find the Pareto optimal set (the Pareto front), whereby each objective is a product of calculation on the vector $x = (x_1, \ldots, x_P)$ of P parameters. In this chapter, optimisation refers specifically to the goal of function minimisation and x equates to the set of locations that make up the track, T.

Determination of Pareto optimal solutions is facilitated by the concept of Pareto dominance. A parameter vector, x_1 is said to strictly dominate a vector, x_2 (denoted by $x_1 \prec x_2$) iff

$$\begin{matrix} \lambda_d(x_1) \leq \lambda_d(x_2) \quad \forall d = 1 \dots D \quad and \\ \lambda_d(x_1) < \lambda_d(x_2) \quad for\ some\ d. \end{matrix} \tag{4.3}$$

Additionally, a vector, x_1 can weakly dominate a vector, x_2 (denoted by $x_1 \preccurlyeq x_2$) iff

$$\lambda_d(x_1) \leq \lambda_d(x_2) \quad \forall d = 1 \dots D. \tag{4.4}$$

A non-dominated set is a set of parameter vectors where no member of the set is dominated by any other member and where every member can therefore be said to be Pareto optimal. The complete set of non-dominated solutions constitutes the Pareto front, $\mathcal{P}$ and can be written as,

$$x_1 \nprec x_2 \quad \forall x_1, x_2 \in \mathcal{P} \tag{4.5}$$

In a single objective system, the selection of a suitable candidate solution at each iteration of the EO algorithm is preceded by a process of ranking the fitness values of each of the mutations from lowest to highest. This can be written as

$$\lambda(x_1) \leq \lambda(x_2) \leq \dots \leq \lambda(x_p) \tag{4.6}$$

where *P* is the number of parameters or in this case, the total number of locations in the track. Incorporation of multiple objectives into this ranking process requires the reduction of the multiple fitness values down to a scalar value. This value can then be used to assess the mutation's position in the rank order relative to the other mutations. The process of Pareto dominance ranking (PDR) (Fonseca & Fleming) achieves this by assessing the count of mutations that strictly dominate a given mutation such that

$$PDR(x_i) \sum_{j=1, j \neq i}^{P} \begin{cases} 1, & x_j \prec x_i \\ 0, & otherwise \end{cases}. \tag{4.7}$$

Once the PDR has been calculated for each mutation, the rank order can be determined using

$$PDR(x_1) \leq PDR(x_2) \leq \dots \leq PDR(x_p). \tag{4.8}$$

Where *PDR* scores are equal, their positions in the rank order are randomly shuffled. This process occurs as part of each iteration and ensures equal

probability of selection when the *PDR* score does not distinguish one solution from another.

4.2.4 Preliminary Equations

All distances and speeds are calculated using spherical trigonometry (Zwillinger 2003; Freitas et al. 2008) such that:

$$D_{i,j} = \frac{60}{1852} \times \frac{180}{\pi} \times \arccos\begin{pmatrix} \sin(lat_i) \times \sin(lat_j) + \cos(lat_j) \\ \times \cos(lat_j) \times \cos(dlon) \end{pmatrix} \quad (4.9)$$

$$S_{i,j} = \frac{D_{i,j}}{\Delta T_{i,j}} \quad (4.10)$$

where distance $D_{i,j}$, between locations i and j, is given in metres. Angles are in radians. lat_i and lat_j are location latitudes of locations i and j respectively. $dlon$ is the difference between longitudes in locations i and j. Speed $S_{i,j}$ is in metres/second and $\Delta T_{i,j}$ is the time in seconds between location i and j. The formula for finding a new location NL from an existing location and a vector of distance and bearing (Williams 2010) is given by:

$$NL_{lat} = \arcsin(\sin(lat) \times \cos(d) + \cos(lat) \times \sin(d) \times \cos(h)) \quad (4.11)$$

$$NL_{lon} = \begin{cases} lon, & \cos(lat) = 0 \\ \mathrm{mod}\left(lon - \arcsin\left(\frac{\sin(h) \times \sin(d)}{\cos(lat)}\right) + \pi, 2 \times \pi\right) - \pi, & \cos(lat) \neq 0 \end{cases} \quad (4.12)$$

where mod(x,y) computes the remainder of x/y and $\pi = 3.1416$.

Argos defines their error measure as one standard deviation of the estimated location error (Hays et al. 2001; Argos 2008). This distance defines the radius of a circle where the centre is marked by the true location and within which we can reasonably expect to find the estimated location 68% of the time. In this chapter I have used empirically determined values for each location class with separate values for latitude and longitude as given by Vincent et al. (2002). The values for the 95%-ile for latitude and longitude were used to produce elliptical error regions about each Argos location. These values are shown in Table **4.1**.

Table 4.1. *Latitudinal and longitudinal location class errors as determined by* Vincent et al. (2002). *These are the values for the 95%-ile of distances from the actual location to the location reported by Argos.*

LC	Latitude ($LC95_{Lat}$)	Longitude ($LC95_{Lon}$)
3	326	742
2	511	1355
1	1265	3498
0	5517	15361
A	5373	10393
B	155356	41219

4.2.5 Optimisation Data Structure

The state of the optimisation system at any given time is represented by a single set of location positions. There are two forms of locations in the system, Argos locations (AL) and interpolated locations (IL). These locations are arranged so that an interpolated location exists between each Argos location in the temporal sequence so that the track, T is given by

$$T = AL_i, IL_i, AL_{i+1}, IL_{i+1}, AL_{i+2}, \ldots IL_{n-1}, AL_n \tag{4.13}$$

where n is the total number of Argos locations. The total number of combined locations is therefore

$$n_T = n \times 2 - 1 \tag{4.14}$$

Each of these locations maintains a set of properties including count, latitude, longitude in decimal degrees and time in seconds. The count starts with a value of 0 and is increased by 1 each time the location is selected from the set of mutated locations. Both of these location types can vary their position in response to the optimisation's mutation and selection processes. This is achieved by modifying the latitude and longitude properties. Additionally, the Argos location type also retains an initial location state which is never modified and is set by the actual reported Argos location. The Argos location also stores the location class (LC) error bounds for the major and minor axis of the error ellipse (Table **4.1**) so that

$$AL_i = \begin{cases} count, & number\ of\ times\ selected \\ init\_lat, & actual\ Argos\ latitude \\ init\_lon, & actual\ Argos\ longitude \\ lat, & adjustable\ latitude \\ lon, & adjustable\ longitude \\ time, & as\ given\ by\ Argos\ system \\ lc\_lat & LC95_{lat}\ for\ LC_i \\ lc\_lon & LC95_{lon}\ for\ LC_i \end{cases} \tag{4.15}$$

The interpolated location, IL_i, has no reference to location class or initial Argos location and is given by,

$$IL_i = \begin{cases} count, & number\ of\ times\ selected \\ lat, & adjustable\ latitude \\ lon, & adjustable\ latitude \\ time, & calculated \end{cases} \tag{4.16}$$

This location represents time not as a stored constant as in AL_i but rather as a calculated value that is determined by its position relative to its adjacent Argos locations such that,

$$timeIL_i = timeAL_i + \frac{(timeAL_{i+1} - timeAL_i) \times \mathrm{GCD}(AL_i, IL_i)}{\mathrm{GCD}(AL_i, IL_i) + \mathrm{GCD}(IL_i, AL_{i+1})} \tag{4.17}$$

where $timeIL_i$ is the time property of the *i*th interpolated location and $timeAL_i$ is the time property of the *i*th Argos location. GCD is the function for calculating great circle distance (Equation 4.9).

4.2.6 Definition of Objectives

The requirement for designing an objective for the EO algorithm is to define two different scalar measures of fitness. The first returns a value that represents the current state of the system and is here on defined with the symbol, λ. The second function provides the optimiser with a relative measure of the effect of a given mutation on the overall fitness of the system as returned by λ. This function is denoted with the prefix, $\mu\lambda$.

Three objective functions were designed to operate simultaneously with the aim of producing a large set of potential pseudo-locations. The composite effect of these objectives working within the multi-objective framework is to continuously explore the potential space that an animal could travel within given the known Argos locations, their time of acquisition and the potential errors as defined by location class. The design of the objectives is such that the probability of a particular mutation being selected is directly related to the probability of that

behaviour occurring. In other words, if the data supplied to the system dictates that travelling at a certain speed or through a given space is unlikely, then it will follow that this behaviour will have a lower chance of being represented in the results. Detailed descriptions of each of these objectives are given in the following sections.

4.2.6.1 Population Speed Objective

The population speed probability distribution function (PDF) is a measure of the distribution of speeds of a representative population that can be used to guide the interpolation structure of an individual animal. In this chapter, Argos location data from 49 individual Antarctic fur seals were used. The combined number of locations after being filtered at 3 m/s for all tracks was 5333. All speeds from consecutive location pairs were grouped together and used to generate a histogram with $n_{bins} = 50$ equally spaced bins (Figure **4.2**). The bin width was chosen qualitatively so as to give a reasonable overall distribution shape. The speed resolution of the bins in this PDF is therefore 3.0 / 50 = 0.06 m/s. The probability (i.e. height) for each bin is denoted by PS_i while the lower and upper speed bounds for each bin are referred to as LS_i and US_i, respectively.

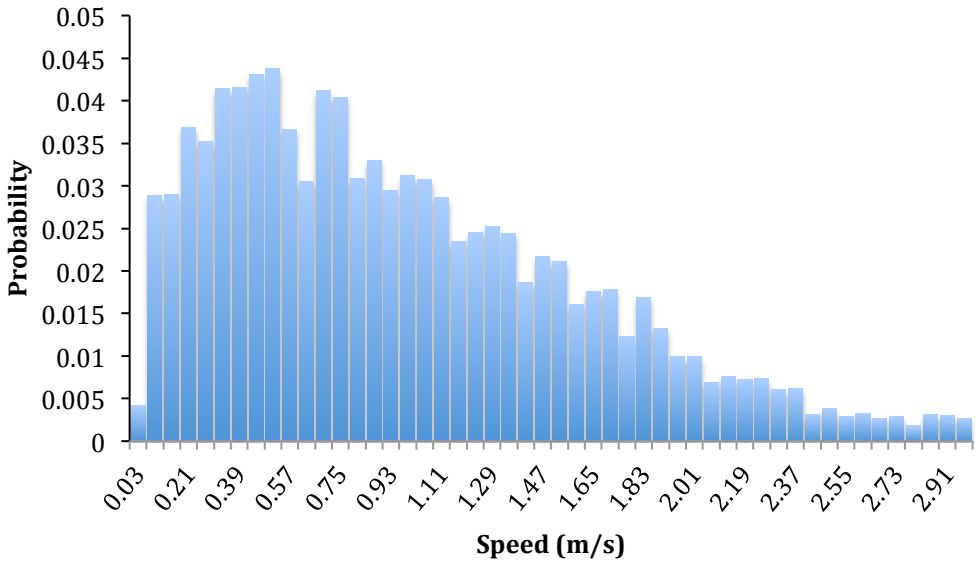

Figure 4.2. *Probability density function of location pair speeds for all tracked Antarctic fur seals.*

At each iteration of the optimiser algorithm, the set of pseudo-locations and their pair-wise speeds is compared to the predetermined population level PDF

through the use of the chi-squared goodness-of-fit, χ^2 (Zar 2009). The calculations for overall fitness of the system are as follows. The total duration of the track in seconds, T_{dur} is found using

$$T_{dur} = timeAL_n - timeAL_1 \tag{4.18}$$

where $timeAL_1$ and $timeAL_n$ are the acquisition times of the first and last locations in the track. The expected value in seconds for the χ^2 for each bin, E_i is given by

$$E_i = PS_i \times T_{dur}, \quad i = 1 \dots n_{bins}. \tag{4.19}$$

The observed value, O_i is the total amount of time of all track sections across all iterations of the EO process that have a speed that is within the upper and lower bounds of the bin, i. The value for each bin is determined using

$$O_i = \frac{\sum_{j=1}^{n_{iters}} \sum_{k=1}^{n_T - 1} \begin{cases} timeL_{k+1} - timeL_k, & LS_i < S_{k,k+1} \leq US_i \\ 0, & otherwise \end{cases}}{n_{iters}}, \quad i = 1 \dots n_{bins} \tag{4.20}$$

where $timeL_k$ is the time of location, k and $timeL_{k+1}$ is the time of the next location, $k+1$, regardless of location type (Argos or interpolated). The number of iterations that EO has run so far is given by n_{iters} and the total number of locations in the track by n_T. The value of the current fitness, λ_{PSO} is then found with

$$\lambda_{PSO} = \sum_{i=1}^{n_{bins}} \frac{(O_i - E_i)^2}{E_i}. \tag{4.21}$$

The mutation fitness of this objective is determined by recalculating the fitness value, λ_{PSO} with the mutated location inserted, denoted as μ_{PSO} and finding the difference such that

$$\mu\lambda_{PSO} = \mu_{PSO} - \lambda_{PSO}. \tag{4.22}$$

This ensures that a negative value of $\mu\lambda_{PSO}$ represents a beneficial mutation and a positive value indicates a deletarious mutation. Where the mutation has no significant effect on the overall fitness of the system, $\mu\lambda_{PSO}$ will always be 0.

It should be noted here that these equations are conceptualised representations which if implemented directly would be extremely computationally inefficient. The actual software coding of this objective involved

the caching of many interim results and moving average calculations, resulting in a 100 fold improvement in calculation times.

4.2.6.2 Maximum Speed Objective

The purpose of the maximum speed objective is to ensure that the set of postulated interpolated locations do not include unreasonably fast track sections. This objective works by informing the optimising algorithm when sections of a track are causing excessive speeds. At the same time, sections that fall below the maximum speed have no influence on the behaviour of the system. This is achieved by clamping parts of the equation so that all negative values are treated as zero. The clamp zero function (cz) is defined as:

$$cz(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{4.23}$$

The speed fitness calculation for a track is given by:

$$\lambda_{speed} = \sum_{i=1}^{i=n-1} cz\left(S_{i,i+1} - M\right) \tag{4.24}$$

where $S_{i,i+1}$ is the speed from location i to the next location in the track and M is the maximum speed constant set by the user. The effect of mutation of a location position l_i on λ_{speed} is calculated as follows:

$$\mu\lambda_{speed}(l_i) = \begin{matrix} \lambda_{speed} + cz\left(\mu S_{i,i-1} - M\right) - cz\left(S_{i,i-1} - M\right) \\ +cz\left(\mu S_{i,i+1} - M\right) - cz\left(S_{i,i+1} - M\right) \end{matrix} \tag{4.25}$$

where $\mu S_{i,i+1}$ is the speed from the mutated location i to the next location in the track and $\mu S_{i,i+1}$ is the speed from the previous location.

In practice, the value for M is chosen to coincide with the speed that the data was initially filtered at (3.0 m/s in this case). When used in this way, it plays no part in the selection process when a location's speed is below the maximum, thereby permitting the population speed objective to control the system. However, when mutations are produced that result in excessively high speeds, this objective acts to guide the system back to a reasonable set of travel speeds that fall within the range of the speed PDF function.

4.2.6.3 Uniform Selection Objective

This objective serves to even out the distribution of selected locations across the spatial extent of the track thereby ensuring that the potential spatial and

temporal variation is thoroughly explored. Without this objective, the system has a tendency to under-represent high speed sections of the track. This is a simple objective that works by keeping track of the number of times each location is selected and returning a fitness measure based on these values. The equation for overall fitness is given by

$$\lambda_{uniform} = \sum_{i=1}^{n_T} countL_i \tag{4.26}$$

and represents the mean number of selected locations. The mutated fitness is simply the selection count for that location such that,

$$\mu\lambda_{uniform} = countL_i. \tag{4.27}$$

4.2.7 Mutation Operator

The mutation operator has two forms, one for Argos locations and one for interpolated locations. The Argos mutation operator selects a random location from within a uniform elliptical distribution where the semi-major and semi-minor axes are determined by the latitudinal and longitudinal error sizes in Table **4.1**. The shape of this distribution is not indicative of the distribution of Argos errors which is known to be non-uniform (Vincent et al. 2002). By selecting a uniform distribution, the search algorithm is able to fully explore all possible combinations of locations with equal probability. It is only after the addition of the influences provided by the input data and the system objectives that a final probability distribution is created. The mutation operator is defined as,

$$rd = \sqrt{\text{rand}()} \tag{4.28}$$

where *rd* is a random distance and rand() is a uniform random number generator returning a value between 0 and 1.

$$rh = \text{rand}() \times 2 \times \pi \tag{4.29}$$

where *rh* is a random heading such that *rd* and *rh* form the polar coordinate of a point sampled from a uniform circular distribution.

$$rd_x = LC95_{lat} \times rd \times \sin(rh) \tag{4.30}$$

$$rd_y = LC95_{lon} \times rd \times \cos(rh) \tag{4.31}$$

such that rd_x and rd_y are euclidean coordinates of a uniform elliptical distribution with axes determined by $LC95_{lat}$ and $LC95_{lon}$. These are converted to polar coordinates,

$$\mu d = \sqrt{rd_{lon}^2 + rd_{lat}^2} \tag{4.32}$$

$$\mu h = \begin{cases} \text{atan}\,(rd_{lat}/rd_{lon}), & rd_{lat} > 0 \\ \pi/2, & rd_{lat} = 0 \end{cases} \tag{4.33}$$

where μd and μh are polar coordinates of a uniform elliptical distribution. Therefore $\mu L_i$is the mutated form of location L_i such that,

$$\mu AL_i = \text{NL}(AL_i, \mu d, \mu h) \tag{4.34}$$

where NL is the new location function that uses equations 5.4 and 5.5 to generate a new position from the addition of the vector defined by μd_i and μh_i to the Argos location L_i.

The Argos mutation operator contains no amount of auto-correlation between successive updates and is therefore a pure random number within an elliptical boundary. Mutation of the ILs is not constrained by a predefined boundary but instead dynamically updates the size of its random steps in response to its temporal and spatial position relative to its adjacent Argos locations. Furthermore, these steps are added to the location's current position thereby making this a form of random walk. The reason for modifying the step size is to allow for finer scale adjustments when travelling speed between the IL's adjacent ALs is approaching the maximum allowable speed of the animal. In such a case, large changes in position are more likely to be rejected as the resultant IL position would probably cause an excessive travel speed. Determining the random walk step size requires the calculation of a mutation factor,

$$MF_i = maxSpeed \times (timeAL_{i+1} - timeAL_i) - 0.8 \times \text{GCD}(AL_i, AL_{i+1}) \tag{4.35}$$

where *maxSpeed* is the maximum speed chosen for the initial speed filtering of the location data (see Section 4.2.2) and $timeAL_{i+1}$, $timeAL_i$ are the times of the ALs before and after IL_i. $\text{GCD}(AL_i, AL_{i+1})$ is the shortest possible distance between the two ALs. The closer this distance is to the distance the animal could travel if moving at the maximum speed, the smaller the value of MF_i gets. The multiplier of 0.8 was used to enforce a small amount of mutation even when the

shortest distance is equal to the maximum possible distance. This value was determined experimentally and was found to provide good search characteristics at both high and low speed sections of a track.

The mutation factor is then applied to a random uniform circular distribution such that

$$\mu d_i = \sqrt{\text{rand}()} \times MF_i \tag{4.36}$$

$$\mu h = \text{rand}() \times 2 \times \pi \tag{4.37}$$

and a new mutated location is created by applying this vector to the current location using

$$\mu IL_i = \text{NL}(IL_i, \mu d_i, \mu h). \tag{4.38}$$

4.2.8 Equi-temporal Location Sampling

At the conclusion of each iteration of the EO algorithm, the current state of the system is recorded. This state represents a track of potential locations that the animal could have travelled between. Interpretation of these locations is made using a technique similar to that of Chapter 2. A set of pseudo-locations (PL) is generated by calculating equi-temporally spaced PLs along a linearly-interpolated path that intersects each location in the track. In order to ensure a uniform spread of the pseudo-locations between EO iterations, the starting point of the first location is chosen at random at the start of each iteration using

$$timePL_1 = \text{rand} \times SF. \tag{4.39}$$

Here, $timePL_1$ is the time in seconds of the first PL, rand is a function that returns a random number between 0 and 1 from a uniform distribution and SF is the temporal sampling frequency. The value of SF was set to 3600 seconds (1 hour). This was determined experimentally and was found to give a thorough and adequate representation of the changing state of the system when viewed across all iterations of the EO. All subsequent PLs were temporally spaced SF seconds apart starting at $timePL_1$ such that

$$timePL_i = timePL_1 + i \times SF \tag{4.40}$$

where $timePL_i$ is the time of *i*th PL. The geographic coordinates of PL_i are determined by first finding the nearest track location before the PL and the nearest location after the PL. These are termed L_a and L_b respectively. The time of these

locations is then used to calculate the normalised distance, ND_i which is the proportional distance that the PL will be placed, relative to its two adjacent track locations, assuming linear interpolation. This is given by

$$ND_i = \frac{timePL_i - timeL_a}{timeL_b - timeL_a}. \tag{4.41}$$

The PL latitude and longitude are then calculated using

$$\begin{aligned} latPL &= latL_a + (latL_b - latL_a) \times ND_i \\ lonPL &= lonL_a + (lonL_b - lonL_a) \times ND_i. \end{aligned} \tag{4.42}$$

The benefit of this approach is that over the course of 1000s of iterations, all track sections are sampled with equal probability. Additionally, since each pseudo-location represents the same temporal window, the spatial distribution of these pseudo-locations can translate directly into a density function (see Section 4.2.11).

4.2.9 Optimising Interpolation Algorithm Processing Steps

The implementation of the EO based interpolation algorithm is summarised as follows:

1. Initialise the track *T*, with the locations supplied by the Argos system and with an interpolated location positioned between each Argos location. The starting latitude and longitude coordinates of the interpolated location is exactly halfway between the coordinates of its adjacent Argos locations. Therefore the system always begins from a state of linear interpolation.
2. For each location L_i in track *T*:
   2.1. Generate a mutated location by applying equation 4.34 or 4.38.
   2.2. Calculate and store the objective fitness scores ($\mu\lambda_{PSO}$, $\mu\lambda_{speed}$ and $\mu\lambda_{uniform}$) for the track given the mutated location.
   2.3. Reverse the mutation so as to return *T* to its state prior to the mutation.
3. Rank the mutation fitness values lowest to highest using PDR (Equation)4.7
4. Select a location from the PDR sorted list of mutations based on the random power law distribution (Equation 4.2). Here, τ was set to 2.2 (see below). Update the track, *T* to reflect this new location.

5. Record a set of pseudo locations based on the current state of T using equi-temporal location sampling (Section 4.2.8).
6. Repeat steps 2 to 5 for 5000 (see below) iterations.

Trials of steps 2 to 4 across several tracks showed that after a rapid exponential reduction, the fitness measures would for most runs, stabilise to a constant level by approximately 3000 iterations. Occasionally this number would increase by a few hundred iterations so it was decided to set the number of iterations at step 6 to 5000 to ensure that the system would always have enough time to settle to its next stable state. Another factor in the performance of the system is the value of τ which controls the search behaviour of the system. When manipulated in conjunction with the number of iterations, the values of $\tau = 2.2$ and 5000 iterations was found experimentally to provide a suitable balance between fully exploring the spatial distribution of the track and the amount of time required to run the algorithm.

4.2.10 Regular Grid Quantisation of Interpolated Pseudo-data

The output of the EO interpolation is a large set of PLs, each of which has equal temporal weighting and whose total spatial distribution is expected to approximate the UD of the satellite tracked animal. The number of PLs is usually very large. For example, if the duration of a track is 1 week and the PLs are spaced every hour, then 1 iteration of the system will record 168 PLs. Therefore, a run of 5000 iterations will result in 840,000 PLs. To generate a UD through the application of a KDE that compares every location to every other location would require enormous computational resources. To reduce the number of data points prior to the application of the KDE, I first quantise the spatial density of the PLs by counting the number of locations that fall within the bounds of a grid cell and assign this count to a single location at the centre of the cell. This is repeated for each cell in a regularly spaced grid to form an unsmoothed density map. The origin of the map in geographic coordinates is given by O_{lat} and O_{lon} and the size of each grid cell (width and height) by CS. The origin, cell size and number of cells in each plane, X and Y, are calculated so that the grid fully encompasses the extent of the PLs. The latitudinal and longitudinal lower and upper bounds of each grid cell, $UM_{x,y}$ is therefore determined by

$$\begin{aligned} latL_y &= O_{lat} + (y-1)\times CS \\ lonL_x &= O_{lon} + (x-1)\times CS \\ latU_y &= O_{lat} + y\times CS \\ lonU_x &= O_{lon} + x\times CS \end{aligned} \quad . \tag{4.43}$$

and the grid is then generated using

$$UM_{x,y} = \sum_{i=1}^{n_{PL}} \begin{cases} 1, & if\ \left(latL_y < latPL_i < latU_y\right) \wedge \left(lonL_y < lonPL_i < lonU_y\right) \\ 0, & otherwise \end{cases} \tag{4.44}$$

to determine the count for each individual cell. This is repeated for all values of $x \in X$ and $y \in Y$.

4.2.11 Kernel Density Estimation of Model Interpolated Data

Smoothing of the quantised pseudo-location density map defined in Section 4.2.10 was achieved using weighted kernel density estimation (see Chapter 3). The location for each data point in the grid was taken to be the centre of the grid cell. The value of the count, $UM_{x,y}$ was used as the weighting parameter. The smoothing parameter was selected using likelihood cross-validation as this was found to be a better estimator of the UD (Chapter 3).

4.2.12 Linear Interpolation

For comparative purposes, a separate linear-interpolated UD was generated in addition to the MIKS UD. Equi-temporal location sampling was applied to the set of non-interpolated Argos locations thereby providing a similar output to the technique detailed in Chapter 2. An unweighted kernel with likelihood cross-validation was used to generate the KDE from the PL data (Chapter 3).

4.2.13 Validation

The precision and accuracy of the MIKS method in predicting the UD was tested using the method described in Chapter 3. The set of speed filtered Argos locations was randomly divided into two equally sized, complimentary tracks. A KDE was generated using MIKS for each of the subset tracks. Bhattacharyya's affinity (BA) and the coefficient of variation of the root mean square error (CV(RMSE)) were used to compare the similarity and difference of the subset UDs (Chapter 3). The consistency of the results across different random subset tracks was evaluated by repeating this process 100 times. Additionally, the quality of these results was determined by creating KDEs using three other alternative approaches. Unweighted and time-weighted KDEs were generated

using the technique in Chapter 3. Also, linear interpolation through the use of equi-temporal location sampling (see Section 4.2.12) was included.

Line plots and parallel plots that highlight the paired nature of the tests were used to visualise the BA and CV(RMSE) results for the 100 test runs. A repeated measures analysis of variance (RMANOVA) was applied to determine significant differences.

The system was written entirely in C++ using the software framework described in Chapter 6. All statistical analysis were performed using R (Ripley 2001).

Foraging trip data from two female Antarctic fur seals were used to test the predictive power of the subset data UDs. Tracks were selected that highlighted different patterns of foraging behaviour. The first animal (AFS1) presented a relatively consistent travel rate throughout its trip, interspersed with short foraging periods. The second animal (AFS2) travelled to and from a remote foraging location where it spent the majority of its time.

4.3 Results

4.3.1 Example Maps

Tracks from two individual female fur seals were used to test the efficacy of four methods for generating an animal's UD. The track for the seal, AFS1 contained 115 locations after being filtered at 3 m/s while AFS2 had 80 locations. Speed data used in the population speed objective (Section 4.2.6.1) was obtained from 49 individual seals with a total of 5333 locations.

The UD map of linearly interpolated PLs is shown in Figure **4.1**. When compared to the map created with MIKS (Figure **4.3**), The shape of the distribution is more narrowly defined about the track and is clearly driven by the linear style of interpolation. Also, it has regions of very low probability as a result of high speed travel through these areas. The MIKS map has a broader distribution that more evenly covers the entire range of the track. The linear interpolated map for AFS2 (Figure **4.5**) has very pronounced peak in the top right corner which is caused by the animal spending the majority of the time in this region. A similar effect can be seen for the MIKS map though a greater

proportion of the distribution has been allocated to the travelling sections of the track.

The validation map showing the two randomly generated subsets for AFS1 (Figure **4.2**) illustrates the linear interpolation's sensitivity to the removal of locations. The narrowness of the distribution combined with the very specific placement of the straight line interpolated PLs produces many regions little to no overlap in the complementary UDs. Comparison with the validation of the MIKS UD tests (Figure **4.4**) shows that this technique is more robust to missing data as the general shape of the complimentary UDs is retained. Additionally, there is greater overlap in the distributions due to increased variance in the distributions. The validation map for AFS2 with linear interpolation (Figure **4.6**) shows a similar result for each of the UD subsets as does the MIKS map (Figure **4.8**) though with a greater variance.

4.3.2 Validation Tests

One way repeated measures analysis of variance were used to separately compare the validation test scores for BA and CV(RMSE) in response to KDE type (unweighted, time-weighted, linear interpolated and model interpolated). The results of the 100 validation test runs were also presented with line plots and parallel plots. The ability for the AFS1 sub-sample UDs to predict their complimentary UD can be seen to show a gradual increase in performance across each of the four KDE types starting with unweighted < time-weighted ≈ linear interpolated < MIKS. This was supported by both the CV(RMSE) results (Figure **4.9**) ($F(3) = 431, p < .001$) and the BA results (Figures **4.10**) ($F(3) = 338, p < .001$). The results for AFS2, (Figure **4.11**) ($F(3) = 156, p < .001$) and (Figure **4.12**) ($F(3) = 264, p < .001$) also indicate that MIKS is the best estimator. Unlike AFS1, the linear interpolated KDE exhibits the worst performance for many of the test run results.

The three-dimensional plots of linearly interpolated (Figure 4.13) and model interpolated (Figure 4.14) UDs demonstrate the differences between these techniques. The linearly interpolated distributions exhibits higher peaks (maximum $p = .00317$) with many narrowly focussed, straight sections. The MIKS map has a more spread out distribution with less pronounced peaks (maximum $p = .00145$).

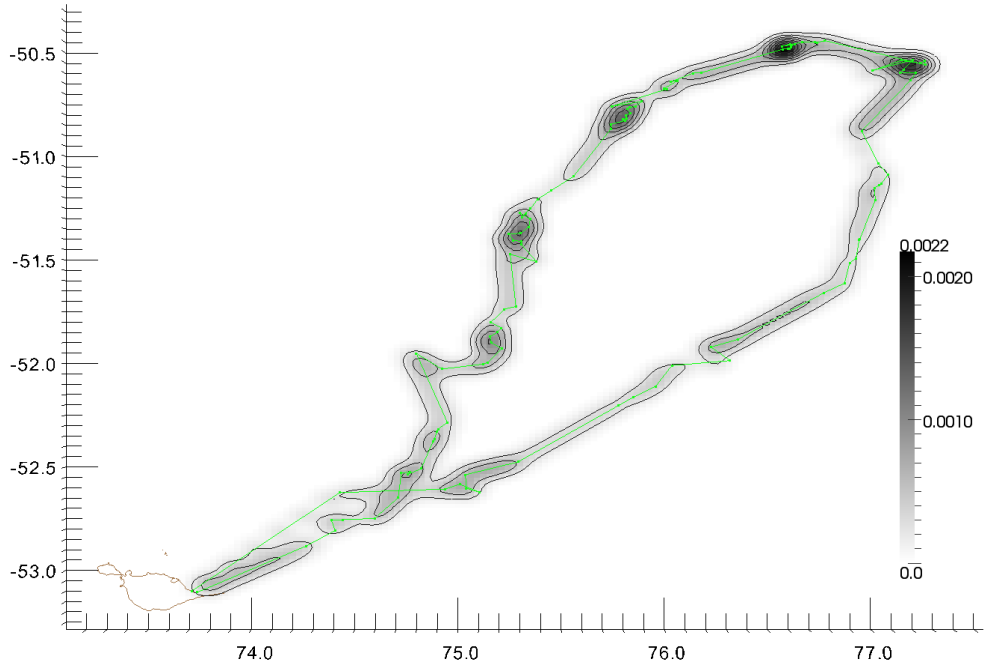

Figure 4.1. *UD of foraging trip, AFS1, generated from a KDE of linearly interpolated Argos locations using likelihood cross-validation. Green points indicate position of Argos locations with consecutive locations connected by a straight line. Grey scale shading represents the normalised probability distribution with 10% contour lines overlaid. Heard Island is outlined in brown in the bottom, left corner.*

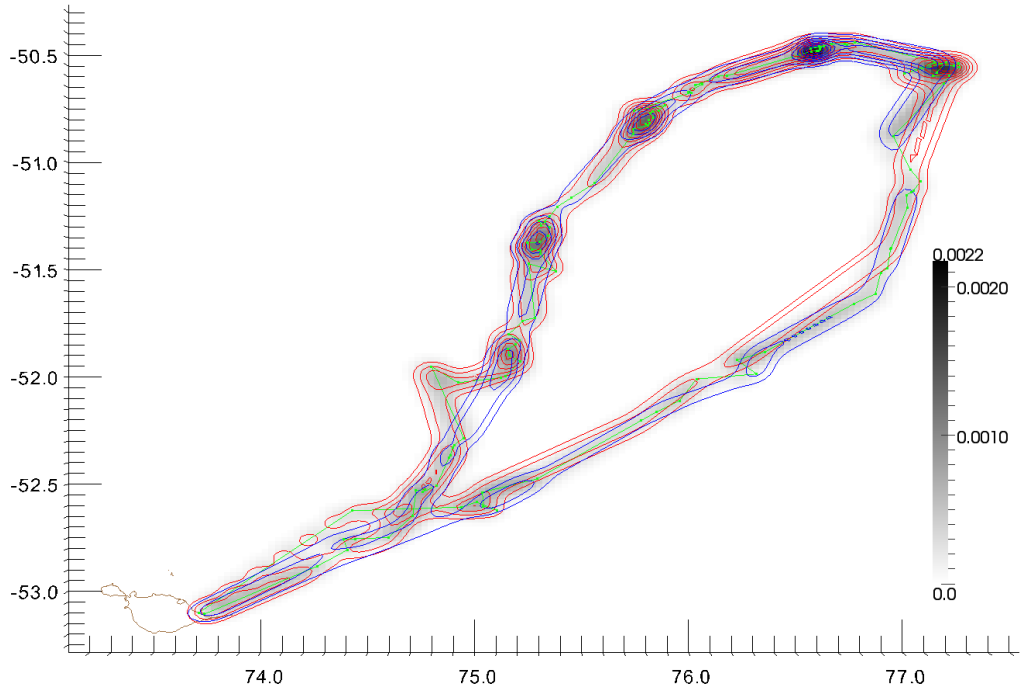

Figure 4.2. *Validation of UDs based on AFS1 subdivided locations with linear interpolation. Red contour lines represent the UD of the first data subset while blue contours are used for the second data subset. Green points and lines define the track of the total set of locations and the grey scale is the UD for all locations.*

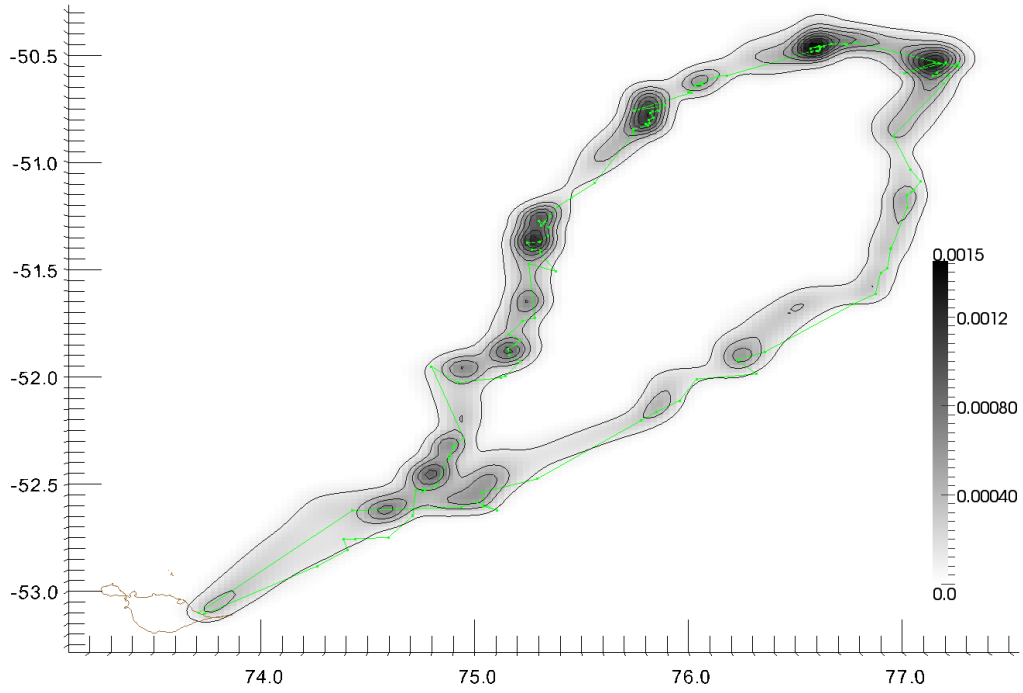

Figure 4.3. *UD of foraging trip, AFS1, using MIKS to smooth Argos locations.*

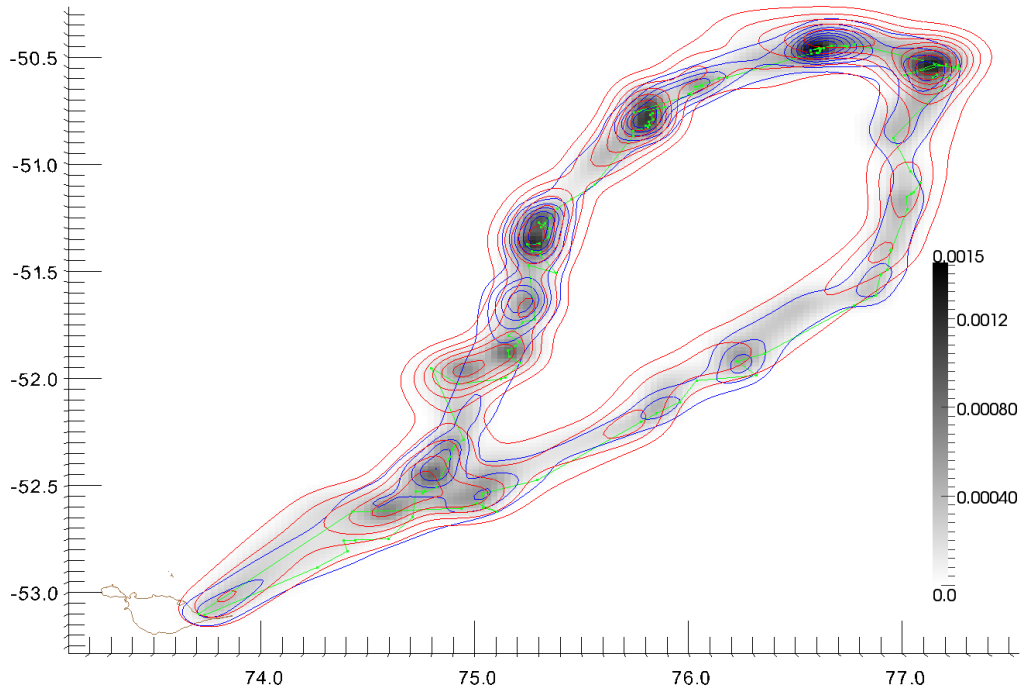

Figure 4.4. *Validation of UDs based on AFS1 subdivided locations with MIKS.*

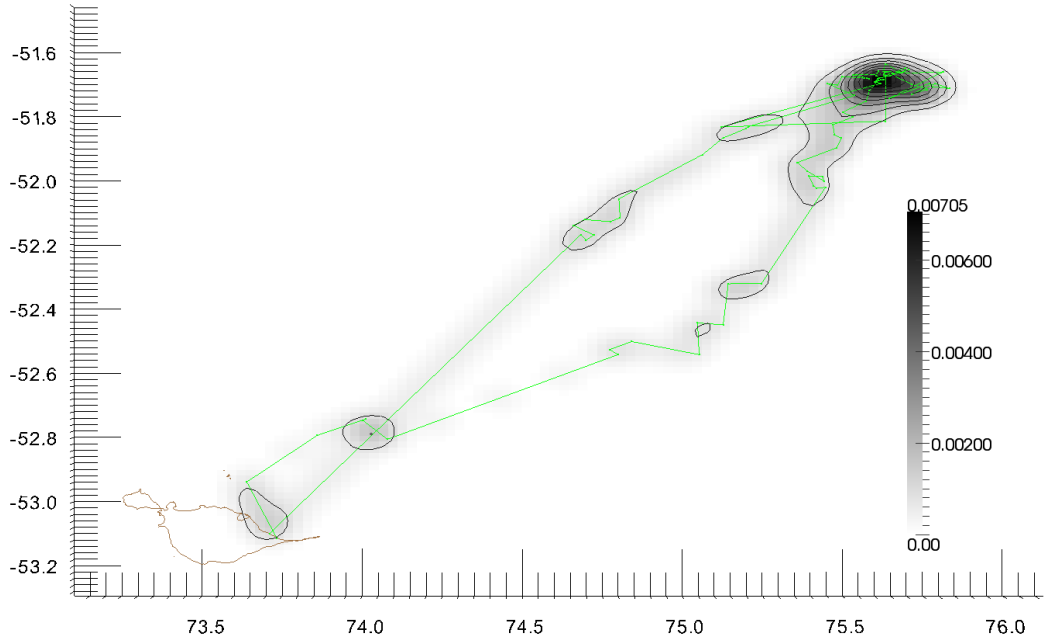

Figure 4.5. *UD of foraging trip, AFS2, generated from a KDE of linearly interpolated Argos locations.*

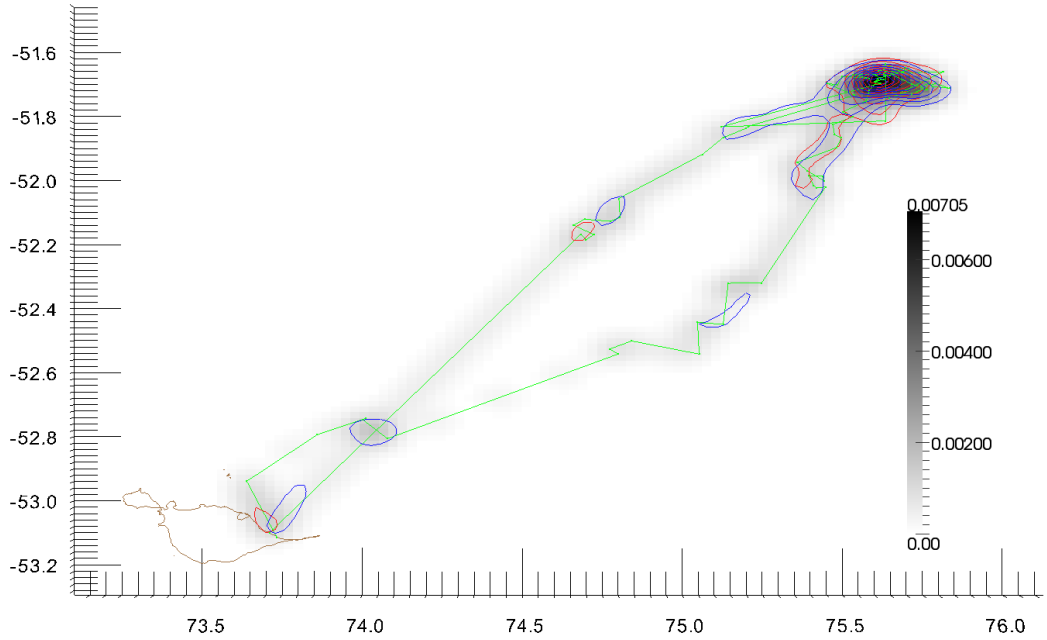

Figure 4.6. *Validation of UDs based on AFS2 subdivided locations with linear interpolation.*

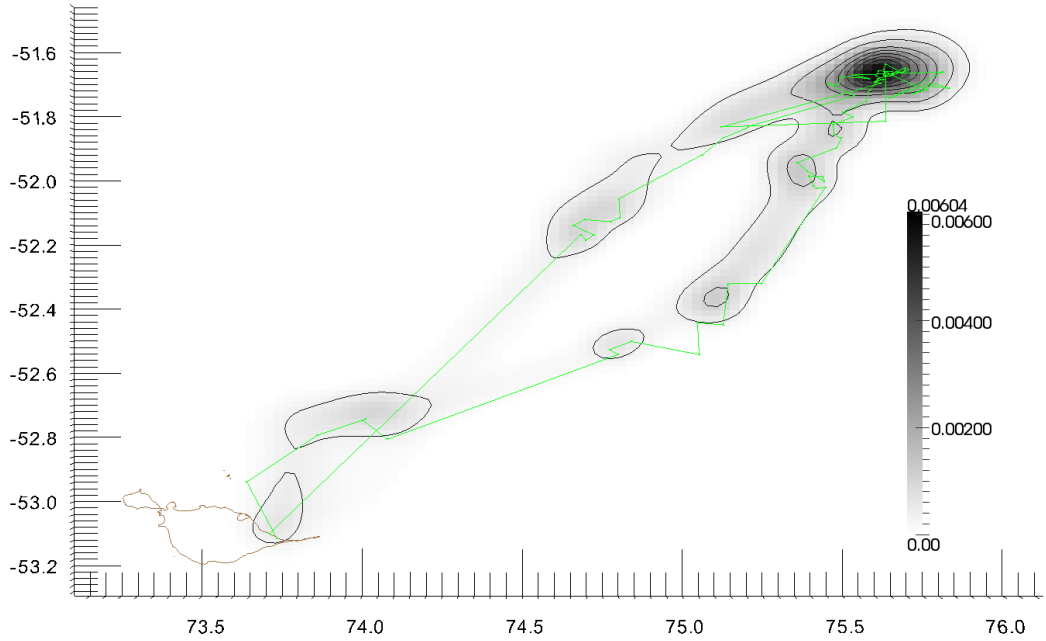

Figure 4.7. *UD of foraging trip, AFS2, using MIKS to smooth Argos locations.*

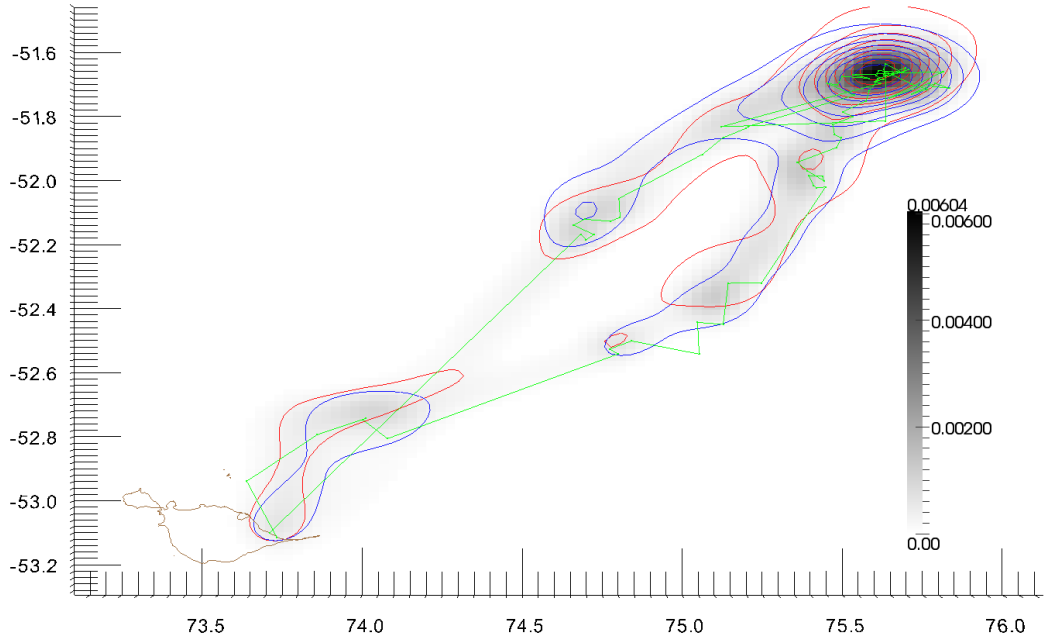

Figure 4.8. *Validation of UDs based on AFS2 subdivided locations with MIKS.*

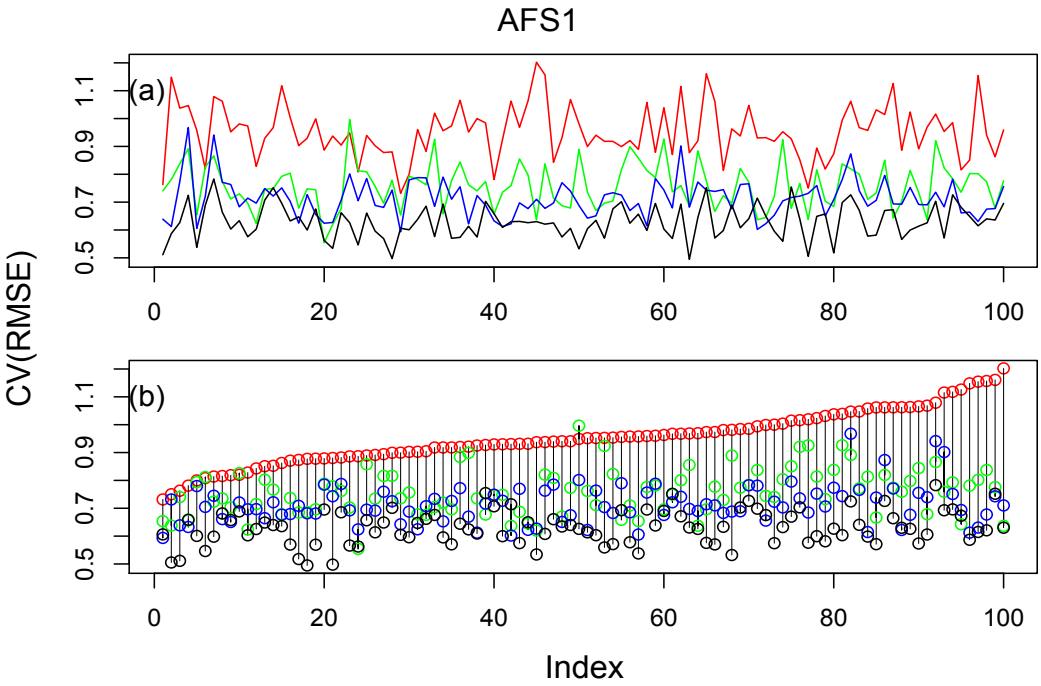

Figure 4.9. *Validation tests of UD subsets repeated 100 times for track, AFS1. Difference between UDs measured by CV(RMSE). Sections (a) shows the raw output values. Section (b) represents the paired nature of this data as parallel plots. Red lines and circles represent unweighted kernel method and green is for the time-weighted kernel. These were included for reference (see Chapter 3). Linear interpolated UD comparisons are shown in blue and MIKS UDs in black.*

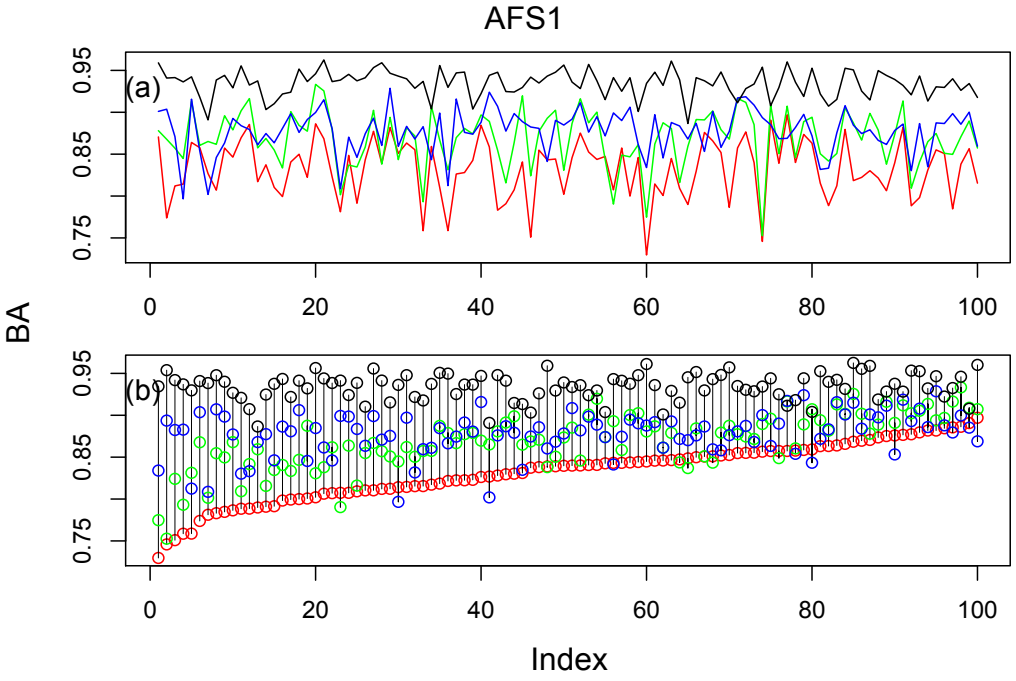

Figure 4.10. *Validation tests of UD subsets for track, AFS2. Similarity measured by BA.*

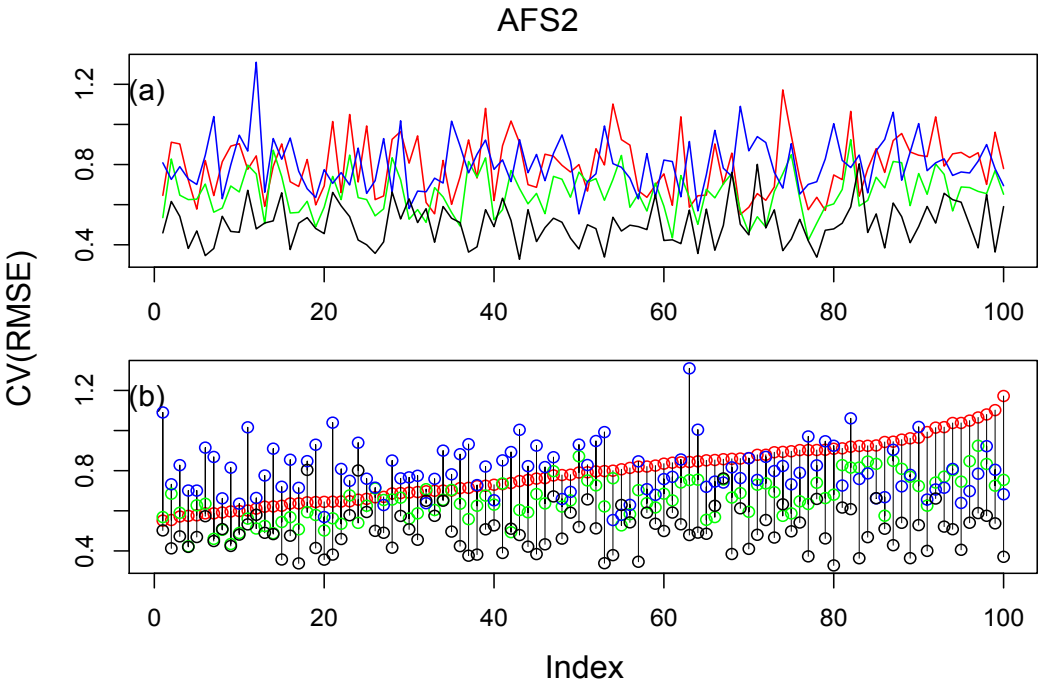

Figure 4.11. *Validation tests of UD subsets for track, AFS1. Difference measured by CV(RMSE).*

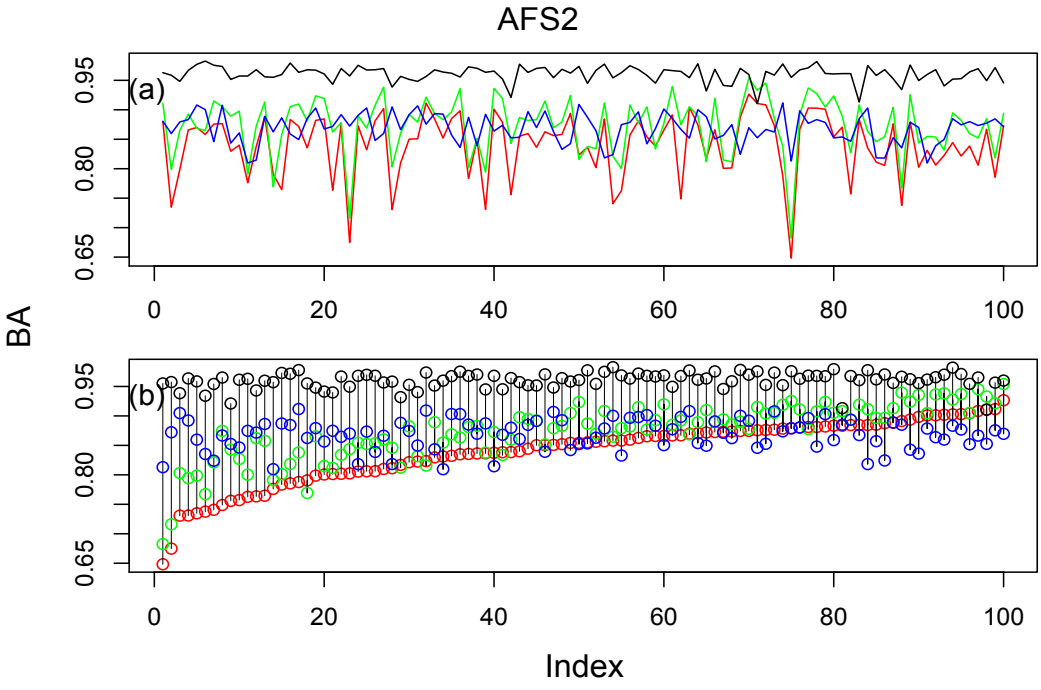

Figure 4.12. *Validation tests of UD subsets for track, AFS1. Similarity measured by BA.*

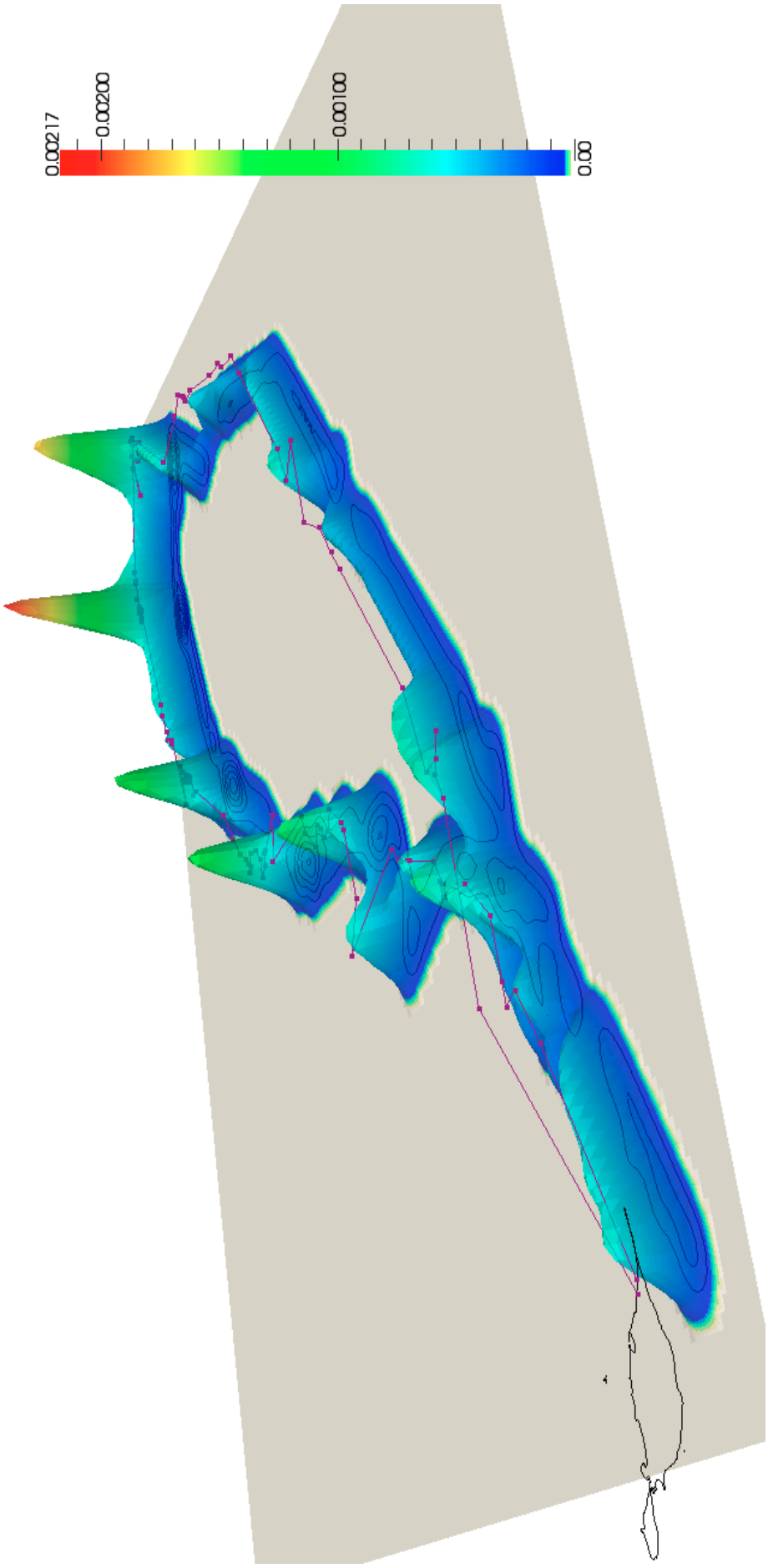

Figure 4.13. Three dimensional representation of linearly interpolated UD from AFS1 locations.

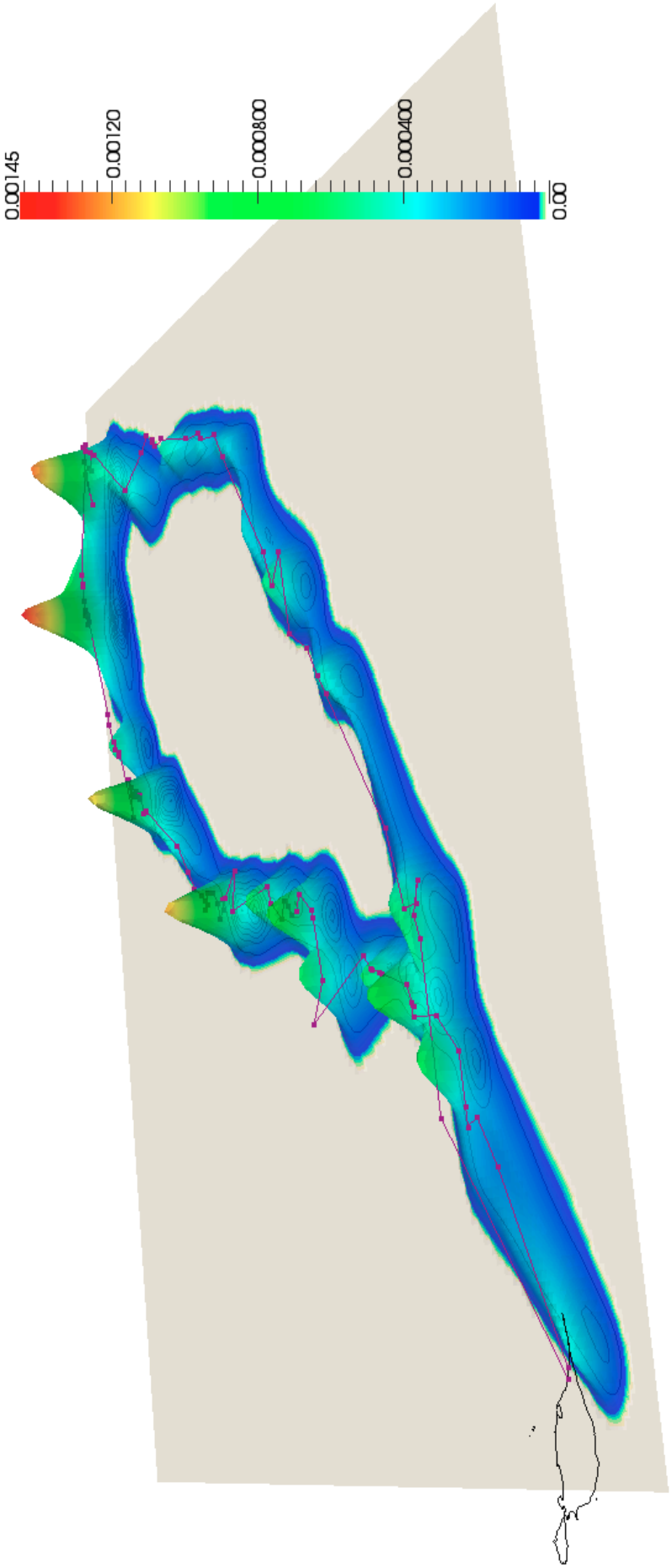

Figure 4.14. Three dimensional representation of a MIKS UD from AFS1 locations.

4.4 Discussion

The utility of sporadic and inaccurate location data in determining patterns of movement behaviour is very much dependant on the quality of the analysis and in particular the ability to extract as much information as possible from a data source with an inherently low signal-to-noise ratio. In this study, the effectiveness in predicting unknown spatial usage through randomised location subsets was assessed for unweighted, time-weighted, linearly interpolated and MIKS UDs. These tests were performed on two different styles of Antarctic fur seal movement tracks. One (AFS1) had a more consistent travel rate with many localised regions of slow movement most likely due to foraging and resting. The other seal (AFS2) travelled out to a foraging location, spent most of its time there and then travelled back. When compared to the simpler methods, MIKS was shown to consistently provide greater overlap with its complimentary validation data set.

Kernel smoothing (KS) has been used in various animal movement studies as a mechanism for the estimation of spatial utilisation (Wood et al. 2000; Matthiopoulos et al. 2004; Phillips et al. 2004; Frydman & Gales 2007). Its effectiveness, however, is limited by issues such as the sample size of the location data set (Matthiopoulos 2003) and the potential error around each data point (Royer & Lutcavage 2008). Appropriate forms of interpolation have been shown to improve the analysis of sparse data sets by estimating intermediate locations (Guinet et al. 2001; Tremblay et al. 2006; Lunardi 2009) and providing a mechanism to regularise the data (Royer & Lutcavage 2008). By combining model driven interpolation with kernel density estimation, MIKS was developed to address these issues. This technique provides a method for KDE that incorporates information about location error and uses empirically determined measures of behaviour to drive interpolation of a limited quantity of infrequently sampled locations. Matthiopoulos (2003) also developed a method for enhancing the standard KS approach with additional information. However, his technique was applied as a composite of a standard KS UD integrated with additional distributions derived from other sources. The method presented here affects the data prior to the application of KS. A logical next step would therefore be to merge these two techniques. A future study could be used to test the relative

performance of KS against MSKS, MIKS and a composite supervised and interpolated KS method.

The results for the AFS1 track showed that linear interpolation out performed the two non-interpolated kernel methods. An interesting finding was the performance of linear interpolation when applied to AFS2. In this case, the effectiveness of this technique was often lower than the non-interpolated methods when compared across multiple validation tests. This is most likely due to the increase in uncertainty that occurs in response to large temporal or spatial gaps in the data (Turchin 1998). When the linearly interpolated travel time is low relative to the expected maximum speed of the animal and no data exists to enforce a particular path of movement then inter-location uncertainty is increased (Royer & Lutcavage 2008). Conversely, as the linear travel time approaches the maximum speed, confidence in the accuracy of linear interpolation will improve. MIKS naturally accounts for this by spreading its distribution out, thereby acknowledging that its predictive ability in these slow moving regions has reduced due to the lack of data. The linear interpolation approach, however, continues to enforce a well defined path of travel, regardless of speed, even though the paucity of local data may not warrant such a distribution. Given the propensity with which linear interpolation gets used in the animal tracking literature, research into the appropriate use of this form of interpolation and its sensitivity to different styles of movement should be assessed further in future studies.

The population distribution of speeds as defined within the speed probability distribution function was determined from a set of experimentally acquired Argos locations. The error contained in these locations therefore has the potential to cause an over-estimation in the calculated inter-location speeds (Austin et al. 2003). Since it has been shown that the size of location error relative to scale of measurement between locations can often approach a negligible level (Lonergan et al. 2009), no attempt was made to quantitatively address this issue. This is, however, a potentially serious issue and one that needs to be addressed in future studies. Fortunately, the availability of more accurate location data from recently developed GPS based technologies such as Fastloc

will; permit the generation of population level distribution functions from locations with a much greater degree of accuracy.

Close examination of the MIKS UD map for AFS1 (Figure **4.3**) shows that the centre of the distribution path does not always coincide with the overlaid straight line joining the Argos locations. This is in contrast to the linearly interpolated UD (Figure **4.1**) where all long stretches of travel have the centre of the probability distribution and the overlaid track closely aligned. This is a product of the elliptical shape of the error surrounding each Argos location fix. Ad hoc testing of MIKS with a circular error distribution showed a much greater alignment to the linear track. Given that the actual error distribution has been experimentally proven to be non-circular (Vincent et al. 2002), this is a potentially important finding. Furthermore, Argos now supplies individual elliptical error estimates on a per location basis which while not being available at the time the data in this chapter was collected, will be incorporated into future versions of the MIKS algorithm. The function of the optimising algorithm is to minimise its objectives which include both maximum speed and population based speed distribution. To achieve this, the algorithm is able to take advantage of the additional potential error in the longitudinal axis of the ellipse by moving the estimate of Argos location into these regions. It is therefore quite possible that these deviations away from the more intuitively acceptable straight line connections are in fact correct. Proof of this would require validation against more accurate data such as that obtained from global positioning system tags. While not in itself proof, the shape of the validation subset UDs may lend some weight to this idea. The validation map (Figure **4.4**) shows two different distributions built from completely different data subsampled from the one track and yet in many places the diversions away from the straight line are in approximate agreement across the two subset UDs. These findings suggest the need for a future study to test the authenticity of these path deviations. If the results of such a study supported the conclusions made here then caution should be exercised before applying linear interpolation to Argos derived location data.

The purpose of this study was to develop an algorithm that could utilise additional sources of information to guide the generation of a probability

distribution estimate of an animal's spatial usage. A number of studies that use SSM and Bayesian statistics (e.g. Jonsen et al. 2003; Patterson et al. 2008) have had a similar aim. The intention here was not to compete with these methods but rather to explore an alternative approach. It would be informative in future studies to make quantitative comparisons between these techniques. Such a comparison would likely inform the validity of assumptions that underlie the modelling structures as well as improve our interpretation of the influence of data distribution, quality and frequency. It may even be possible to combine aspects of each of the methodologies into a larger analytical framework.

Many evolutionary computation techniques exist for finding Pareto optimal sets of multi-objective solutions. These include genetic algorithms (Fonseca & Fleming), particle swarm optimisation (Fieldsend & Singh 2002) and ant colony optimisation (Marco & Gianni 1999). All of these metaheuristic algorithms function by iteratively modifying a population of multiple candidate solutions. The discerning factor is the way that the individual solutions interact with each other to influence successive generations of solutions. By maintaining more than one solution, different regions of a solution space can be simultaneously explored with the intention that over time, they will converge to a single final end result (Osyczka 2002). EO is different in that it only ever has a single solution state which represents an adjustable set of function coefficients (Boettcher & Percus 2001). In designing an approach to mapping a large number of successive location interpolations for the purpose of generating a probability distribution, I was specifically interested in finding a simple multi-objective algorithm whose intermediate states represented a feasible interpolated location track solution and could therefore be recorded as data for subsequent kernel smoothing. While the choice in search algorithm proved successful, it was not without problems. In particular, the design of the mutation operator was a critical component in the algorithm's performance. A more sophisticated mutation operator that modifies its behaviour in response to the success of previous iterations has been proposed (Chen et al. 2007) and its use here could offer improvements to the functionality of this approach. Additionally, there are other single solution metaheuristic techniques such as simulated annealing (Kirkpatrick et al. 1983) that might also perform well in this context and should also be explored.

Several factors serve to confound the process of translating a set of locations into a large scale estimate of an individual's UD. These include the number of locations, the temporal and spatial distribution of those locations and the level of uncertainty or error about each of those locations. In this chapter, I have presented a new form of location interpolation that can be used to estimate a spatial probability distribution. By incorporating knowledge of Argos location error with a population based speed probability distribution, a UD was produced that was shown to better predict spatial usage than other simpler methods.

Chapter 5

A New Method for Filtering Argos Location Data

Abstract

Argos satellite tracking is frequently used to determine the movement of marine animals. Associated with this system is a variable degree of error in the estimation of locations. The magnitude of this error is such that a significant proportion of locations in a typical animal track are biologically unfeasible. A common approach to identifying and removing erroneous locations involves the measurement of speed between locations in order to determine which locations to remove. In this I chapter I propose an alternative form of speed filter that incorporates estimates of location error into a location's potential position. By adjusting location positions within regions defined by their estimated error, this filter was able to remove fewer locations while still satisfying the requirement for not exceeding a maximum speed. The two filters were run against the same sets of data from Antarctic fur seals (*Arctocephalus gazella*), Macaroni penguins (*Eudyptes chrysolophus*) and King penguins (*Aptenodytes patagonicus*). For each set of animals, the filters were run at four speeds, 1, 2, 3 and 4 m/s. With the exception of the 1 m/s test where both filters performed similarly, the new filter consistently removed approximately half the number of locations.

5.1 Introduction

Knowledge of the spatial ecology of marine animals is important for research into areas such as behaviour (Thompson et al. 2003a), physiology (Kuhn et al. 2006), habitat use (Burns et al. 2004), ecosystem dynamics (Skern-Mauritzen et al. 2009) and conservation and management of living resources (Cunningham et al. 2009).

A frequently used technology for tracking marine animals is the combination of animal borne transmitters (e.g. Sirtrack, Wildlife Computers) and the Argos satellite system (Argos 2008). The Argos system relies on a set of

equations that take into account knowledge of the transmitter frequency and movement of the satellite relative to the transmitter to extrapolate location via the effect of Doppler shift on the carrier signal frequency. The complete solution to these equations requires the reception of a minimum of four separate transmitter messages. Argos then provides users with an estimate of location in the form of latitude and longitude, the time of the location and an estimate of error. When less than four locations are received, locations are still able to be estimated, though it is not possible to determine the level of error. Error levels are reported as a data field called a location class (LC) and describe the radius of a circle with its centre at the reported location. The radius of the circle represents the 68% confidence interval for the predicted error rate of the given location class. These classes are LC3 < 250m; 250m < LC2 < 500m; 500m < LC1 < 1500 and LC0 > 1500m (Argos 2008). Additionally there are two classes with indeterminate error known as LCA and LCB. LCZ which is also occasionally reported is considered invalid and is therefore usually discarded. Empirical measurements of LC have not always concurred with these values. In particular, (Vincent et al. 2002) found that LCA had similar accuracy to LC1 and was significantly better than LC0 while LCB had the greatest errors of all LCs. They also found that latitudinal errors were less than Argos predictions while longitudinal errors consistently exceeded Argos predictions.

Recently, an additional, more accurate form of error estimation has been introduced that describes the error as an ellipse defined by size and orientation (Argos 2008). This error information is preferable to location classes as it is more accurate and provides the potential for separate latitudinal and longitudinal estimates as well as being calculated independently for each location. Unfortunately, as this service is not retrospective, pre-existing datasets will need to continue to use LC error estimates.

Many of the studies that utilize Argos location data (Le Boeuf et al. 2000; Boehlert et al. 2001; Campagna et al. 2001) acknowledge the presence of location error but then go on to treat the locations as if they were exact positions (Hays et al. 2001). Recently, a number of studies have started to address this issue by quantitatively incorporating the location error into the analysis through the use of state space models and the Kalman filter (e.g. Jonsen et al. 2003; Thompson et al.

2003b). By incorporating a behaviour based (e.g. transit and foraging) multi-state individual movement model with a model of observational error, state space models can be used to improve the locational estimates in a movement dataset over that of the Argos locations alone (Patterson et al. 2008). A potential disadvantage of these likelihood based approaches is the requirement for *a priori* assumptions of modelled individual behaviour. Additionally, the inherent complexity in design and implementation often requires expert statistical involvement (Patterson et al. 2008).

All Argos location data has some error associated with it. For the purpose of studying animal movements on a scale of 10s or 100s of kilometres, an error of 100s or 1000s of metres is usually acceptable (Vincent et al. 2002). However, it is often the case that some of the locations in a track have an excessive positional error that makes these locations unfeasible and therefore invalid given the physiological and behavioural constraints of the animal. A commonly used method for removing erroneous locations is given by McConnell et al. (1992). This filter is an averaging forward and backward iterative equation that relies on a root mean square measure of speed to identify sections of a track that exceed a given maximum speed. It examines the velocity of each location relative to the two preceding and two successive locations and for each iteration, removes the fastest location until none exceed the given maximum speed. This approach makes no attempt to discriminate between location classes or to incorporate the potential error surrounding locations. In practice, the use of this filter can result in the removal of up to 50% of locations (McConnell et al. 1992; Austin et al. 2003) and can eliminate more accurate LC3, LC2 or LC1 locations by passing potentially false LC0, LCA or LCB locations (Austin et al. 2003).

Rejection filters such as McConnell et al. (1992) have the advantage of being relatively simple to implement and use as well as requiring few *a priori* assumptions apart from the specification of a maximum speed. The limitation with this method comes from measuring distances between Argos locations without reference to the location error (Royer & Lutcavage 2008). By ignoring the potential error in reported locations, this filter can overestimate the actual speed travelled. This effect is magnified where locations are spatially or temporally close together and can result in many feasible locations being rejected

(Austin et al. 2003). Consequently, the applicability of this filter is greatly limited when applied to small datasets (Royer & Lutcavage 2008).

The quantitative inclusion of a location error can be used to change the interpretation of an Argos location from an exact position to an estimated or approximate position. The majority of true positions can therefore be expected to exist somewhere within a region that is centred at the location supplied by Argos and bounded by the location's estimated error. When treated this way, the measurement of distance and therefore speed between two variable locations is also subject to variation. If this potential for variation in speed is incorporated into a rejection filter, then it is possible to retain locations that would otherwise be rejected if speed was measured using fixed locations. The aim of this chapter is to present a new form of speed filter that incorporates location error into its calculations by adjusting the position of Argos locations within the confines of the error bounded location regions. Like McConnell et al. (1992), this filter identifies and removes locations that cause excessive high speed rates of travel. However, this filter can also retain more locations by modifying the position of the original Argos locations. The end result is a set of locations with some locations removed and some with slightly different coordinates to the original Argos location. This set of locations meets the dual objectives of each location falling within its location region and each inter-location travel speed not exceeding the maximum speed. The performance of this filter is evaluated against that of McConnell et al. (1992) by applying these filters at four different speeds to several seal and penguin tracks and comparing the numbers of locations in the resulting tracks.

5.2 Methods

5.2.1 Data Collection

The location data used in this study were obtained from seal and penguin colonies at Heard Island between December 2003 and February 2004. All data was received using the Argos satellite system. Platform terminal transmitters (PTT) manufactured by Sirtrack Limited were attached to individual Antarctic fur seals (*Arctocephalus gazella*) caught at Spit Bay (53° 07′ S, 73° 44′ E). The PTT was glued to the back of the animal using Araldite 2017 epoxy adhesive. The PTT was set to transmit on a continuous duty cycle with a 30 second repetition

rate (Robinson et al. In Prep). King penguins (*Aptenodytes patagonicus*) at Spit Bay were fitted with a Telonics ST-10 PTT. The PTT was attached to the animal's back with Loctite 401 glue and secured with two plastic cable ties. Transmission rate was set to 45 seconds (Wienecke & Robertson 2006). Macaroni penguins (*Eudyptes chrysolophus*) were caught at Capsize Bay (53° 10′ S, 73° 39′ E). Sirtrack PTTs were attached to the back with Loctite 401 and cable ties and set to transmit every 45 seconds (Deagle et al. 2008).

5.2.2 McConnell Filter

This filter was implemented as described in McConnell et al. (1992). It is an iterative algorithm that repeatedly calculates the velocity between adjacent locations. After calculating all location velocities, the fastest location is removed. This process is repeated until all velocities fall below the given maximum speed. The formula used to calculate velocity is given by

$$V_i = \sqrt{\frac{1}{4} \times \sum_{j=-2, j\neq 0}^{j=2} \left(v_{i,j+i}\right)^2} \tag{5.1}$$

5.2.3 Optimising Filter

5.2.3.1 Filter Introduction

An intuitive solution to incorporating location error into a filter is to create an algorithm that can move each Argos location within an estimated error bounded region. By designing this algorithm to repeatedly move locations until the travel time between two locations is less than the maximum speed, a new track with adjusted location positions can be formed. For this to work, the method also needs to ensure that all sections of the track have acceptable speeds and that none of the locations fall outside the bounds of their given location region. This can create a conflict if the location boundary does not permit enough adjustment to shorten the track segment so as to reduce the speed to an acceptable level. When such an event occurs, the algorithm can resolve the issue by removing the location. Given that it is not unusual for a track to have hundreds of locations, finding a set of adjusted locations that meet all of these criteria and with as few location deletions as possible is a non-trivial exercise. Since a location can fall anywhere within its region, there is essentially an infinite number of possible

forms that the filtered track can take. Solving this requires a search algorithm that can optimise a function with hundreds of dimensions (where each location region is a dimension) and within a reasonable time frame given the current power of modern desktop computing.

In the previous chapter (Section 4.2.3), I described the concept of mathematical optimisation and the various evolutionary computation approaches to solving functions. I also introduced the technique called extremal optimisation (EO) and explained how it can escape local minima through the use of a power law distributed selection mechanism. A form of EO was also used in this chapter. An important modification to the algorithm was the introduction of a simpler selection system. Here, selection is based on the location whose mutation causes the greatest improvement in fitness. This was possible since the track structure that is based on *a priori* knowledge of location class error creates a system with a strictly bounded range of possible solutions. Although the system is still multi-modal, it is constrained enough that the magnitude of the mutation operator is large enough to prevent the solution being trapped in local minima. This modification has the benefit of improving the efficiency and processing time of the algorithm.

Here I describe a new form of location speed filter that utilises EO to define a path through a set of location regions and that removes locations that prevent the maximum speed objective from being satisfied. Knowledge of Argos locations and their associated LC error was used to create a mutation operator that is restricted to exploring the space defined by these elliptical regions. Two objective fitness functions were constructed, one to assess the maximum speed of sections of the track and the other to restrict the solution to favour being as close to the original Argos locations as possible.

5.2.3.2 Preliminary Equations

An important component of the optimising filter presented in this study is the quantitative inclusion of location class error. Argos defines their error measure as one standard deviation of the estimated location error (Hays et al. 2001; Argos 2008). This distance defines the radius of a circle where the centre is marked by the true location and within which we can reasonably expect to find the estimated location 68% of the time. In this chapter I have used empirically

determined values for each location class with separate values for latitude and longitude as given by Vincent et al. (2002). The values for the 95%-ile for latitude and longitude were used to produce elliptical error regions about each Argos location. These values are shown in Table **5.1**.

Table 5.1. *Latitudinal and longitudinal location class errors as determined by* Vincent et al. (2002). *These are the values for the 95%-ile of distances from the actual location to the location reported by Argos.*

LC	Latitude ($LC95_{Lat}$)	Longitude ($LC95_{Lon}$)
3	326	742
2	511	1355
1	1265	3498
0	5517	15361
A	5373	10393
B	155356	41219

A track *T*, consisting of *N* unfiltered Argos locations *L*, with a location class *LC*, is used to generate a filtered track *t*, with *n* locations *l*, where $n \leq N$. Each location l_i where $i = 1 \ldots n$, is located within an elliptical region centred at L_i. The semi-major axis of the ellipse is defined by the longitudinal error ($LC95_{lon}$) and the semi-minor axis by the latitudinal error ($LC95_{lat}$) (see Table **5.1** for the actual error values).

All distances and speeds are calculated using spherical trigonometry (Zwillinger 2003; Freitas et al. 2008) such that:

$$D_{i,j} = \frac{60}{1852} \times \frac{180}{\pi} \times \arccos\begin{pmatrix} \sin(lat_i) \times \sin(lat_j) + \cos(lat_j) \\ \times \cos(lat_j) \times \cos(dlon) \end{pmatrix} \quad (5.2)$$

$$S_{i,j} = \frac{D_{i,j}}{\Delta T_{i,j}} \quad (5.3)$$

where distance $D_{i,j}$, between locations *i* and *j*, is given in metres. Angles are in radians. lat_i and lat_j are location latitudes of locations *i* and *j* respectively. *dlon* is the difference between longitudes in locations *i* and *j*. Speed $S_{i,j}$ is in metres/second and $\Delta T_{i,j}$ is the time in seconds between location *i* and *j*. The formula for finding a new location *NL* from an existing location and a vector of distance and bearing (Williams 2010) is given by:

$$NL_{lat} = \arcsin(\sin(lat) \times \cos(d) + \cos(lat) \times \sin(d) \times \cos(h)) \quad (5.4)$$

$$NL_{lon} = \begin{cases} lon, & \cos(lat) = 0 \\ \mathrm{mod}\left(lon - \arcsin\left(\dfrac{\sin(h) \times \sin(d)}{\cos(lat)}\right) + \pi, 2 \times \pi\right) - \pi, & \cos(lat) \neq 0 \end{cases} \quad (5.5)$$

where mod(x,y) computes the remainder of x/y and $\pi = 3.1416$.

5.2.3.3 Definition of Objectives

The purpose of the optimising filter is to minimise objective cost functions that assess maximum speed and deviation from Argos location. The algorithm requires that all objectives supply two forms of fitness calculation. The first form is an overall assessment of the current state of the track solution. These are termed λ_{speed} for maximum speed (Equation 5.7) and $\lambda_{location}$ for Argos location deviation (Equation 5.9). The second form tests the effect of a specific location mutation on the overall track fitness and are termed $\mu\lambda_{speed}$ and $\mu\lambda_{location}$ for mutation maximum speed (Equation 5.8) and mutation Argos location deviation (Equation 5.10) respectively.

The maximum speed objective works by informing the optimising algorithm when sections of a track are causing excessive speeds. At the same time, sections that fall below the maximum speed have no influence on the behaviour of the system. This is achieved by clamping parts of the equation so that all negative values are treated as zero. The clamp zero function (cz) is defined as:

$$cz(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (5.6)$$

The speed fitness calculation for a track is given by:

$$\lambda_{speed} = \sum_{i=1}^{i=n-1} cz\left(S_{i,i+1} - M\right) \quad (5.7)$$

where $S_{i,i+1}$ is the speed from location i to the next location in the track and M is the maximum speed constant set by the user. The effect of mutation of a location position l_i on λ_{speed} is calculated as follows:

$$\mu\lambda_{speed}(l_i) = \begin{array}{c} \lambda_{speed} + cz\left(\mu S_{i,i-1} - M\right) - cz\left(S_{i,i-1} - M\right) \\ +cz\left(\mu S_{i,i+1} - M\right) - cz\left(S_{i,i+1} - M\right) \end{array} \quad (5.8)$$

where $\mu S_{i,i+1}$ is the speed from the mutated location i to the next location in the track and $\mu S_{i,i+1}$ is the speed from the previous location. The sum of all distances

from the adjusted location to the position given by the original Argos location is calculated as:

$$\lambda_{location} = \sum_{i=1}^{i=n} D_{i,ai} \tag{5.9}$$

and the effect of mutation on a specific location is:

$$\mu\lambda_{location} = \lambda_{location} + \mu D_{i,ai} - D_{i,ai} \tag{5.10}$$

where $\mu D_{i,ai}$ is the distance from the mutated position to the original Argos location.

5.2.3.4 Mutation Operator

The mutation operator is a function that selects a random location from within a uniform elliptical distribution where the semi-major and semi-minor axes are determined by the latitudinal and longitudinal error sizes in Table **5.1**. The mutation operator is defined as,

$$rd = \sqrt{\text{rand()}} \tag{5.11}$$

where *rd* is a random distance and rand() is a uniform random number generator returning a value between 0 and 1.

$$rh = \text{rand()} \times 2 \times \pi \tag{5.12}$$

where *rh* is a random heading such that *rd* and *rh* form the polar coordinate of a point sampled from a uniform circular distribution.

$$rd_x = LC95_{lat} \times rd \times \sin(rh) \tag{5.13}$$

$$rd_y = LC95_{lon} \times rd \times \cos(rh) \tag{5.14}$$

such that rd_x and rd_y are euclidean coordinates of a uniform elliptical distribution with axes determined by $LC95_{lat}$ and $LC95_{lon}$. These are converted to polar coordinates,

$$\mu d = \sqrt{rd_{lon}^2 + rd_{lat}^2} \tag{5.15}$$

$$\mu h = \begin{cases} \text{atan}\,(rd_{lat}/rd_{lon}), & rd_{lat} > 0 \\ \pi/2, & rd_{lat} = 0 \end{cases} \tag{5.16}$$

where μd and μh are polar coordinates of a uniform elliptical distribution. Therefore $\mu L_i$is the mutated form of location L_i such that,

$$\mu L_i = \mathrm{NL}(L_i, \mu d_i, \mu h_i) \tag{5.17}$$

where NL is the new location function that uses equations 5.4 and 5.5 to generate a new position from the addition of the vector defined by μd_i and μh_i to the Argos location L_i.

5.2.3.5 Identification of High Speed Locations

When a section of a track between two locations is found to have an excessive speed, it could be either location that is responsible. Since one of those locations will be removed, it is necessary to identify which is causing the high speed travel rate. The high speed location (HSL) is found by comparing the speeds before and after the two locations:

$$HSL(L_i, L_j) = \begin{cases} L_i, & S_{i-1,j} > S_{j,j+1} \\ L_j, & S_{i-1,j} < S_{j,j+1} \end{cases} \tag{5.18}$$

where L_i and L_j are the locations between which the speed of travel, $S_{i,j}$ is calculated.

5.2.3.6 Processing Steps in Operation of Speed Filter

The implementation of the speed filter optimising algorithm is as follows:

1. Initialise the track *T*, with the locations supplied by the Argos system.
2. For each location L_i in track *T*:
   2.1. Generate a mutated location by applying equation 5.17.
   2.2. Calculate and store the objective fitness scores ($\mu\lambda_{speed}$ and $\mu\lambda_{location}$) for the track given the mutated location.
   2.3. Reverse the mutation so as to return *T* to its state prior to the mutation.
3. Rank the mutation fitness values from lowest to highest:
   3.1. Order by $\mu\lambda_{speed}$ first.
   3.2. If $\mu\lambda_{speed}$ is equal for locations *i* and *j* then order by $\mu\lambda_{location}$.
   3.3. If $\mu\lambda_{location}$ is equal for locations *i* and *j* then choose with equal probability to either leave as is or swap.
4. Select the first location mutation in the rank list as the mutation to keep and update the track T to reflect this new location.

5. Repeat steps 2 to 4 for 5000 (see below) iterations.
6. Find the fastest section in the track.
   6.1. If the speed between locations is found to be greater than *M*, the maximum speed, then remove the location before or after as decided by equation 5.18.
7. Repeat steps 5 to 6 until all track sections have speeds less than *M*.

When sorting fitness rankings, the algorithm prioritises the maximum speed function over the Argos location function. This can be seen in the sorting routine defined at step 3 where $\mu\lambda_{location}$ is only assessed if $\mu\lambda_{speed}$ is found to be equal. This is a simplified approach to multi-objective assessment that was found to do a good job of enforcing the algorithm's search for a track that obeys the maximum speed constraint.

As in Chapter 4, trials of steps 2 to 4 were conducted across several tracks and found to produce a stable solution by about 3000 iterations. In order to ensure that a stable state was always reached, the number of iterations was increased to 5000.

5.2.3.7 Filter Performance Analysis

Both the McConnell filter and the optimising filter were applied to tracks obtained from 49 Antarctic fur seals, 64 Macaroni penguins and 50 King penguins. The filters were run at four speeds, 1 m/s, 2 m/s, 3 m/s and 4 m/s. Both filters were run at each of the four speeds for all three species. For each speed tested, a boxplot and a two way ANOVA were used to compare the number of locations removed for each filter and for the effect of species on filter performance. A three way ANOVA with filter, species and speed as main effects was also performed.

5.3 Results

During the period between 19 December 2003 and 17 February 2004, a total of 39815 locations (not counting discarded LCZ) were collected from Antarctic fur seals, Macaroni penguins and King penguins. There were 49 individual fur seal tracks with 6507 locations (Table **5.2**); 64 Macaroni penguin tracks with 10670 locations (Table **5.3**) and 50 King penguin tracks with 22638 locations

(Table **5.4**). Of these locations, the greatest proportion were LC1 for Macaroni penguins (30%, n = 3190) (Figure **5.1**b, Table **5.3**) and King penguins (29%, n = 6677) (Figure **5.1**c, Table **5.4**) and LCB for Antarctic fur seals (29%, n = 1889) (Figure **5.1**a, Table **5.2**). However, the distribution of LCs for all locations acquired against each of the three species (Figure **5.1**) was not significantly different ($\chi^2 = 12.95$, df = 10, $p = 0.2264$).

For all three species, the McConnell filter removed approximately twice the number of locations as the optimising filter when run at speeds between 2 and 4 m/s (Tables **5.2**, **5.3** and **5.4**). At 1 m/s both filters performed similarly, removing approximately 50% of the data. When run at 1 m/s, the optimising filter tended to remove slightly more LC3, LC2 and LC1 locations than the McConnell filter while removing fewer of the LC0, LCA and LCB locations. This trend was similar for all three species. At higher speeds, the filters performed similarly for LC3, LC2 and LC1 while the McConnell filter removed substantially more of the LC0, LCA and LCB locations (Figure **5.2**).

A three way ANOVA of Filter, Species and Speed was performed and found that there were highly significant differences within all main effects and first level interactions with only the Filter:Species:Speed interaction showing no significant difference (Table **5.9**). Closer inspection of the behaviour of the filters at different speeds was made using two way ANOVAs with filter type and species as main effects. These were performed for each of the four speeds (1, 2, 3 and 4 m/s) that the filters were run at (Tables **5.5**, **5.6**, **5.7** and **5.8**). Boxplots of these percentage removed data were also presented (Figures **5.3**, **5.4**, **5.5** and **5.6**). At 1 m/s there was no significant difference between the filters performance though there was a significant species effect ($p = 0.026$). This was attributable to a slightly higher rejection rate and much smaller variance in the percentage locations removed for King penguins (Figure **5.3**). At all speeds greater than 1 m/s, there was a significant difference between the two filters ($p < 0.001$) (Tables **5.6**, **5.7** and **5.8**). Differences for species and filter:species interaction were less noticeable. There was a slight difference in the species effect at 4 m/s (F = 3.047, df = 2, $p = 0.049$) (Table **5.8**) and filter:species interaction at 3 m/s (F = 3.076, df = 2, $p = 0.048$) (Table **5.7**).

Table 5.2. *Numbers and percentages of locations acquired for Antarctic fur seals and the numbers and percentages of locations removed by the McConnell and Optimising filters. Results for the two filters are given for speeds from 1 m/s to 4 m/s. Results are broken down by LC as well as their totals.*

	m/s	3	2	1	0	A	B	Total
Unfiltered								
n acquired	-	441	658	1182	1189	1148	1889	6507
% acquired	-	7	10	18	18	18	29	100
McConnell Filter								
n removed	1.0	88	180	447	625	685	1167	3192
	2.0	24	61	181	349	369	612	1596
	3.0	15	37	131	256	284	451	1174
	4.0	11	25	104	218	231	370	959
% removed	1.0	20	27	38	53	60	62	49
	2.0	5	9	15	29	32	32	25
	3.0	3	6	11	22	25	24	18
	4.0	2	4	9	18	20	20	15
Optimising Filter								
n removed	1.0	105	242	499	563	614	1005	3028
	2.0	33	67	132	191	229	307	959
	3.0	19	43	79	125	152	189	607
	4.0	13	31	65	92	111	159	471
% removed	1.0	24	37	42	47	53	53	47
	2.0	7	10	11	16	20	16	15
	3.0	4	7	7	11	13	10	9
	4.0	3	5	5	8	10	8	7

Table 5.3. *Numbers and percentages of locations acquired for Macaroni penguins and the numbers and percentages of locations removed by the McConnell and Optimising filters. Results for the two filters are given for speeds from 1 m/s to 4 m/s. Results are broken down by LC as well as their totals.*

	m/s	3	2	1	0	A	B	Total
Unfiltered								
n acquired	-	590	1789	3190	1930	1337	1834	10670
% acquired	-	6	17	30	18	13	17	100
McConnell Filter								
n removed	1.0	99	465	1248	1254	912	1355	5333
	2.0	30	161	477	778	529	786	2761
	3.0	18	98	325	608	420	618	2087
	4.0	15	77	253	516	364	516	1741
% removed	1.0	17	26	39	65	68	74	50
	2.0	5	9	15	40	40	43	26
	3.0	3	5	10	32	31	34	20
	4.0	3	4	8	27	27	28	16
Optimising Filter								
n removed	1.0	163	714	1381	936	729	961	4884
	2.0	50	186	353	305	247	277	1418
	3.0	37	129	262	212	200	199	1039
	4.0	29	107	201	160	165	150	812
% removed	1.0	28	40	43	48	55	52	46
	2.0	8	10	11	16	18	15	13
	3.0	6	7	8	11	15	11	10
	4.0	5	6	6	8	12	8	8

Table 5.4. *Numbers and percentages of locations acquired for King penguins and the numbers and percentages of locations removed by the McConnell and Optimising filters. Results for the two filters are given for speeds from 1 m/s to 4 m/s. Results are broken down by LC as well as their totals.*

	m/s	3	2	1	0	A	B	Total
Unfiltered								
n acquired	-	927	3055	6677	5260	3207	3512	22638
% acquired	-	4	13	29	23	14	16	100
McConnell Filter								
n removed	1.0	187	946	2754	3153	2425	2789	12254
	2.0	55	318	1002	1833	1399	1677	6284
	3.0	27	196	655	1399	1085	1310	4672
	4.0	22	145	511	1156	927	1076	3837
% removed	1.0	19	33	42	63	72	75	54
	2.0	6	10	15	35	44	48	28
	3.0	3	6	10	27	34	37	21
	4.0	2	5	8	22	29	31	17
Optimising Filter								
n removed	1.0	324	1199	3034	2656	2003	2048	11263
	2.0	78	375	808	847	688	634	3430
	3.0	47	248	522	551	491	379	2238
	4.0	32	191	425	441	393	292	1774
% removed	1.0	33	41	46	53	59	55	50
	2.0	8	12	12	16	21	18	15
	3.0	5	8	8	10	15	11	10
	4.0	3	6	6	8	12	8	8

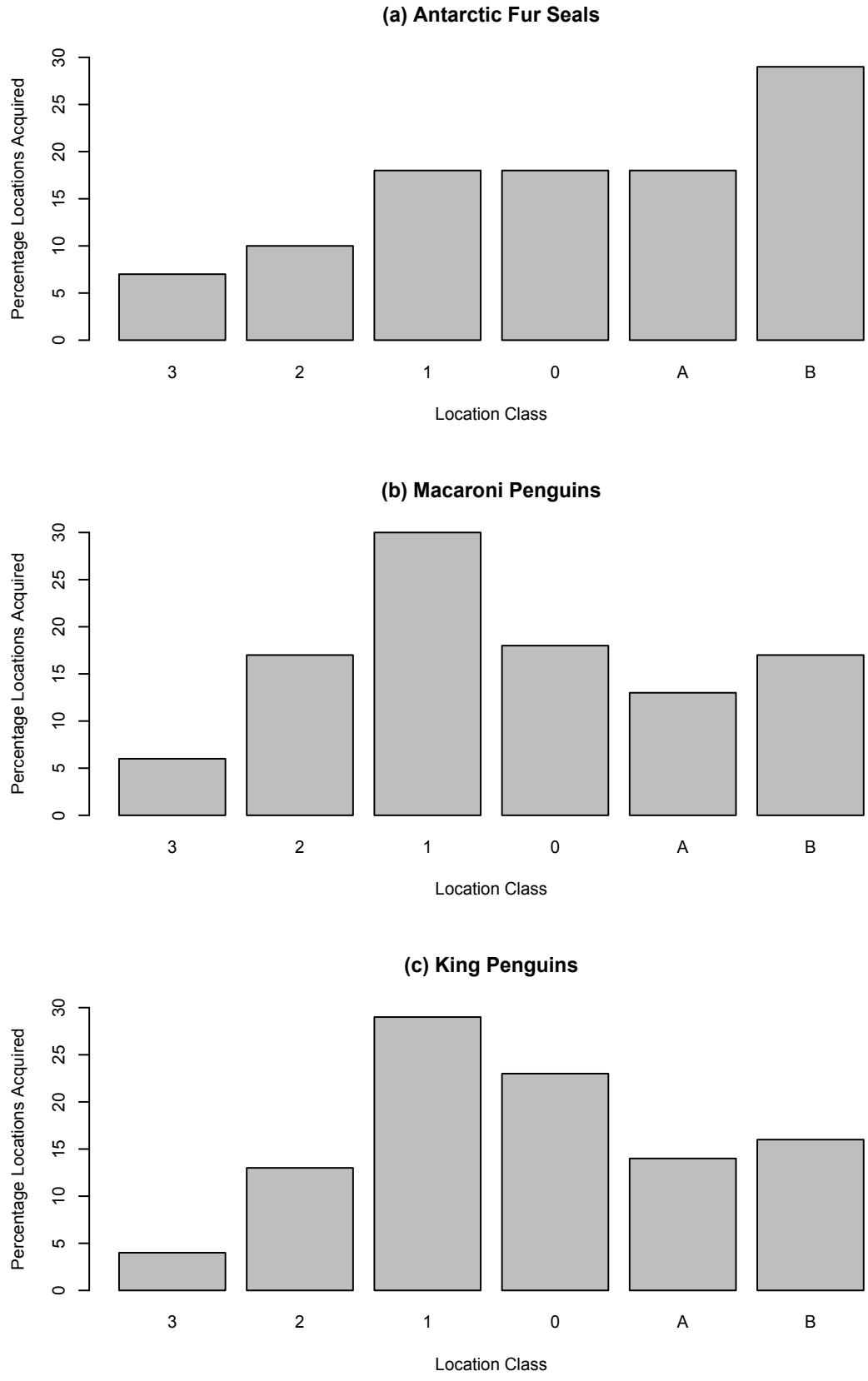

Figure 5.1. *Percentage of locations acquired for each location class. Total number of locations obtained for each species were a) Antarctic fur seals, n = 6507; b) Macaroni penguins, n = 10670; c) King penguins, n = 22638.*

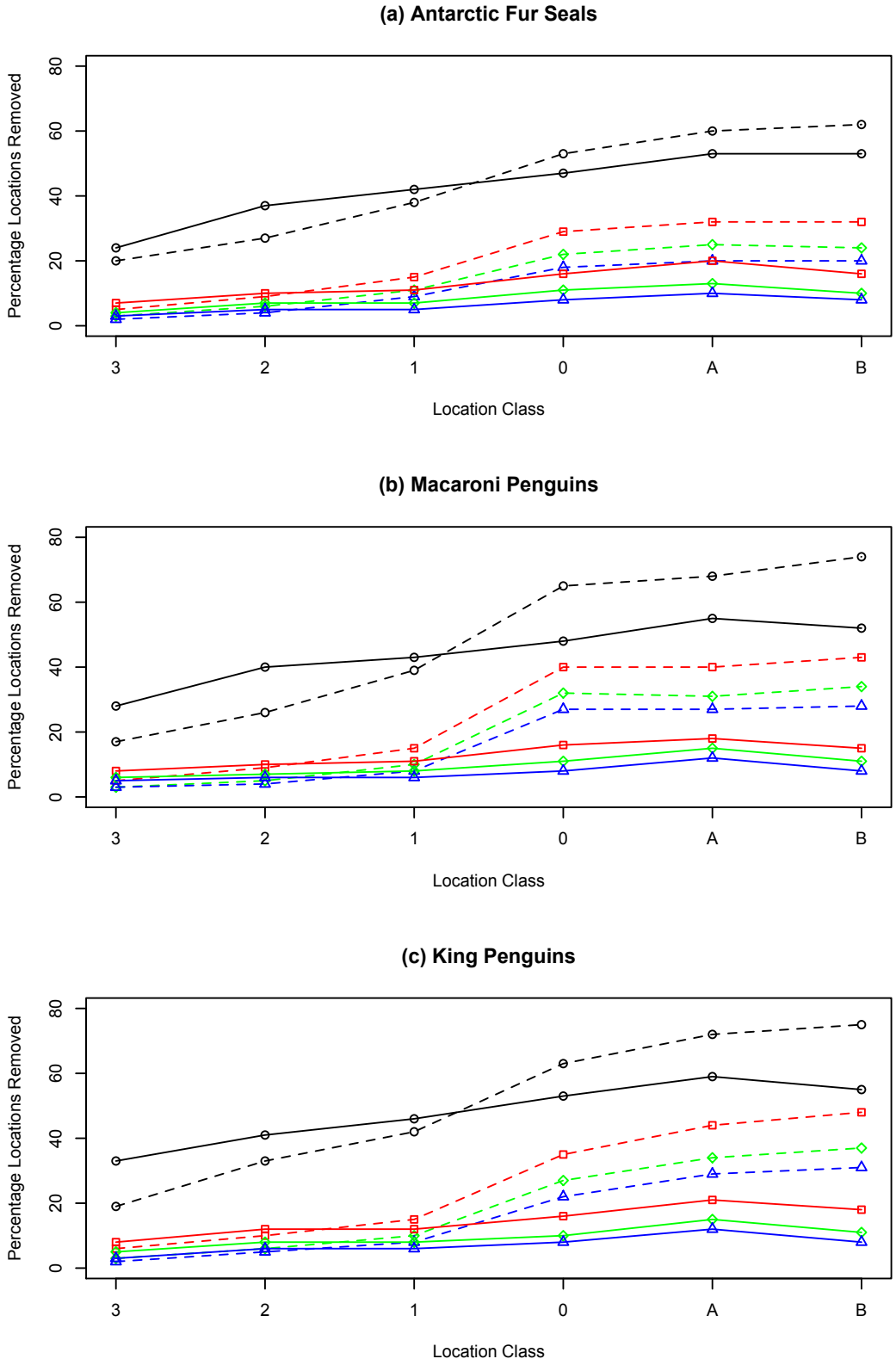

Figure 5.2. *Line plot comparison of percentage locations removed by each filter. Location classes are separated across the x-axis. Plot (a) shows results for Antarctic fur seals, (b) Macaroni penguins and (c) King penguins. Solid lines indicate the optimising filter while the dashed lines represent the McConnell filter. Black circles are for filters run at 1 m/s, red squares for 2 m/s, green diamonds for 3 m/s and blue triangles for 4 m/s.*

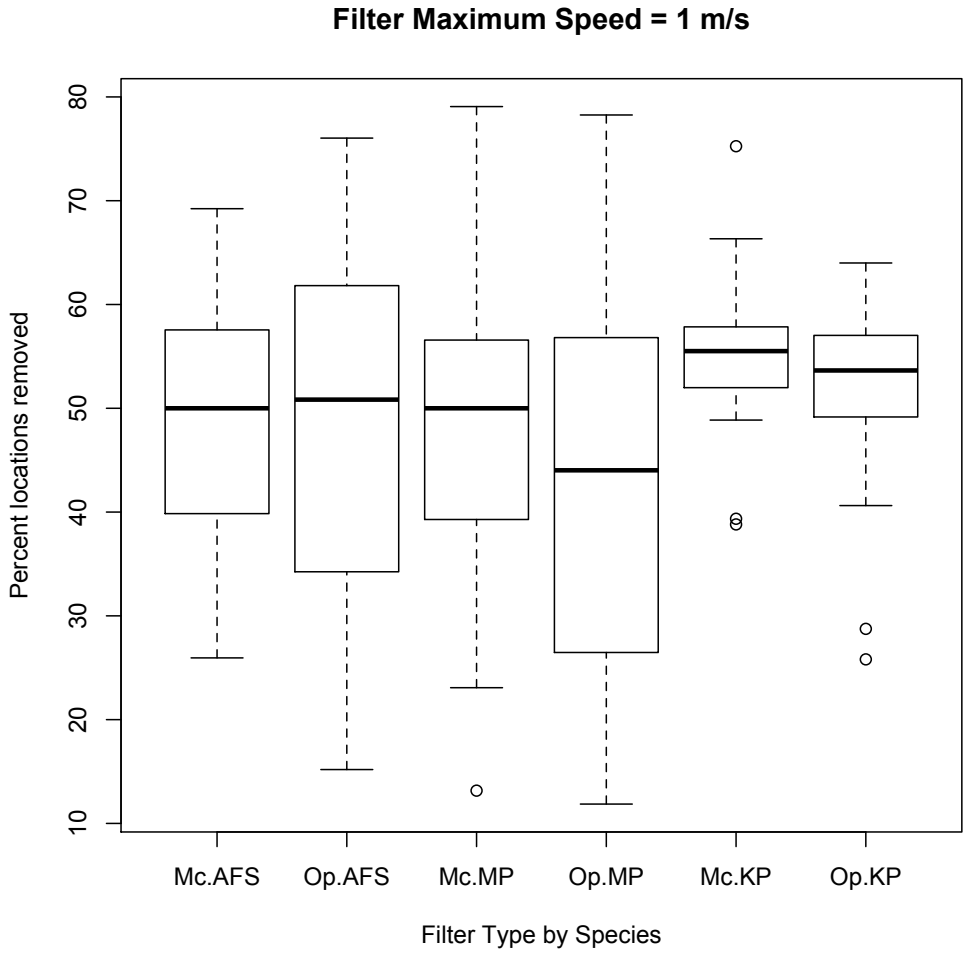

Figure 5.3. *Boxplot of percentage locations removed by McConnell filter (Mc) and Optimising filter (Op) at maximum speed of 1 m/s for each species; Antarctic Fur Seal (AFS), Macaronic Penguin (MP) and King Penguin (KP). Thick line in centre of box indicates the median, top and bottom of box is the upper and lower quartile and the range of the data is shown by the whiskers.*

Table 5.5. *ANOVA of percentage locations removed with a maximum speed of 1 m/s. Main effects are Filter and Species with the interaction effect indicated by the row, Filter:Species.*

	df	SS	MS	F	p
Filter	1	542	542.48	2.7407	0.09907
Species	2	1465	732.71	3.7018	0.02603
Filter:Species	2	218	108.80	0.5497	0.57782
Residuals	252	49879	197.93		

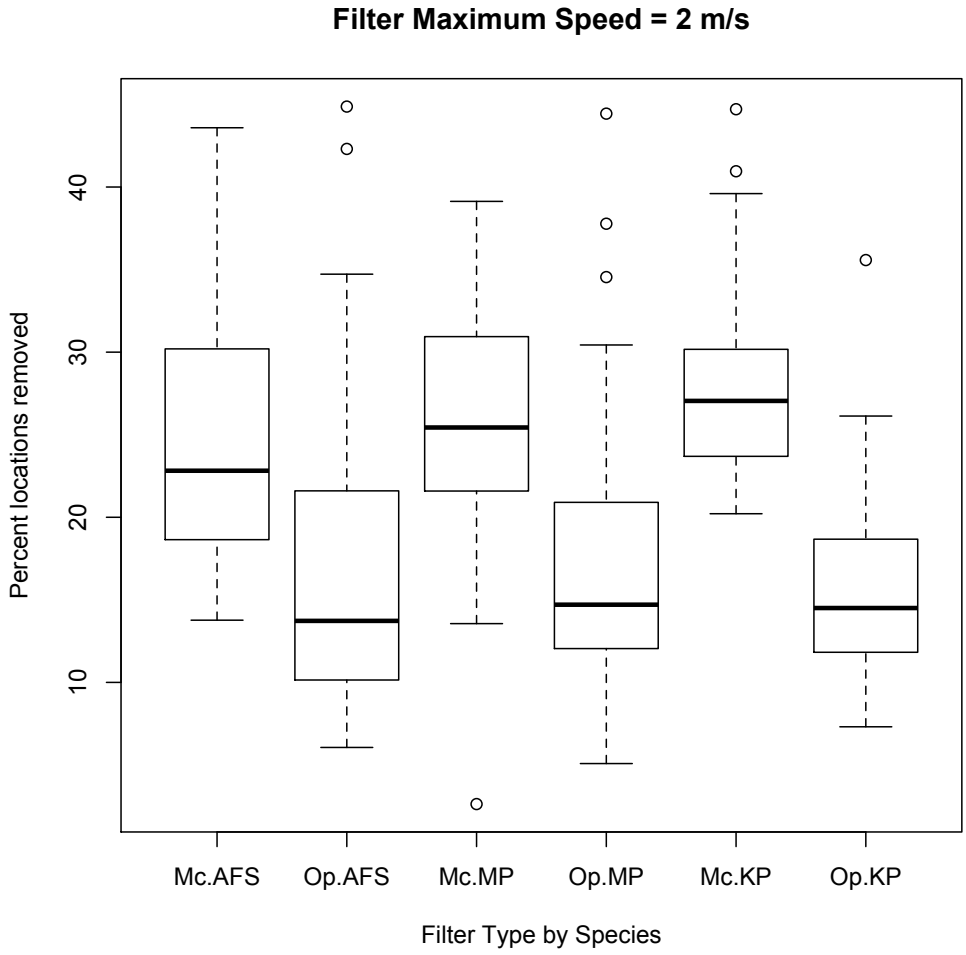

Figure 5.4. *Boxplot of percentage locations removed by McConnell filter (Mc) and Optimising filter (Op) at maximum speed of 2 m/s for each species; Antarctic Fur Seal (AFS), Macaronic Penguin (MP) and King Penguin (KP). Thick line in centre of box indicates the median, top and bottom of box is the upper and lower quartile and the range of the data is shown by the whiskers.*

Table 5.6. *ANOVA of percentage locations removed with a maximum speed of 2 m/s. Main effects are Filter and Species with the interaction effect indicated by the row, Filter:Species.*

	df	SS	MS	F	p
Filter	1	7208.9	7208.9	137.7534	< 2e-16
Species	2	39.7	19.8	0.3792	0.6847
Filter:Species	2	259.1	129.6	2.4760	0.0857
Residuals	317	16589.1	52.3		

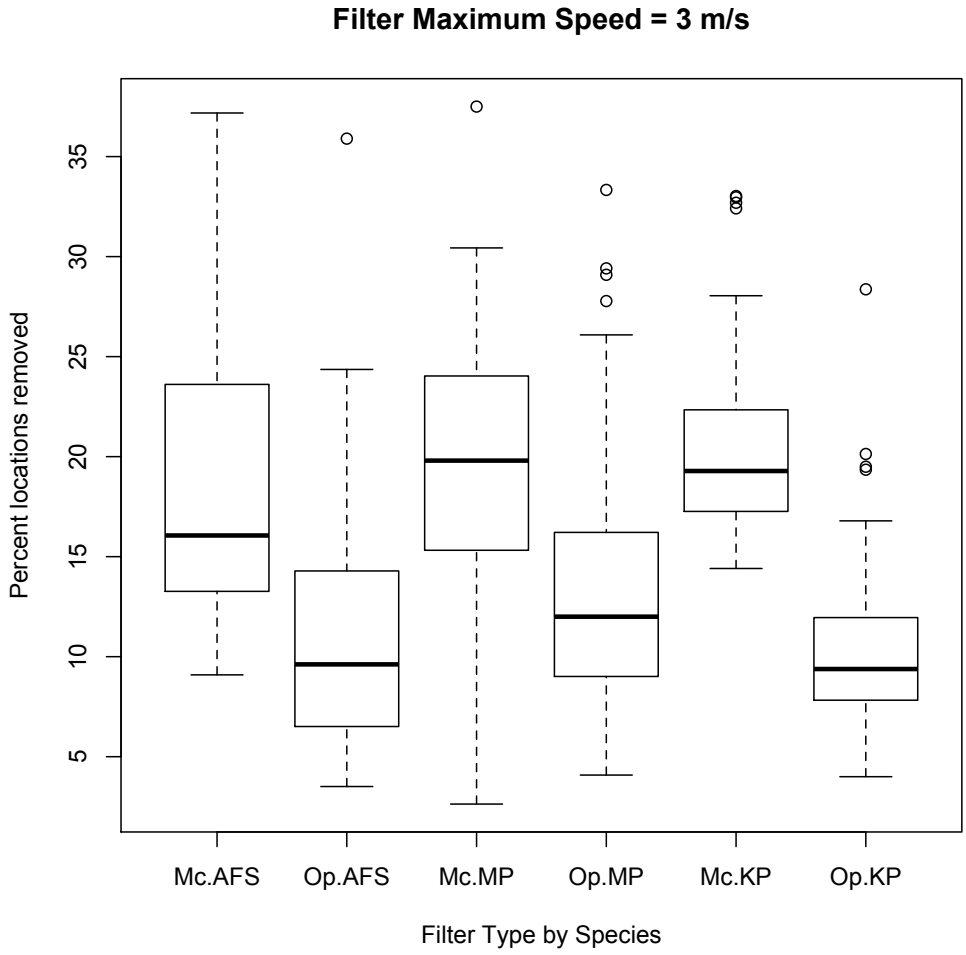

Figure 5.5. *Boxplot of percentage locations removed by McConnell filter (Mc) and Optimising filter (Op) at maximum speed of 3 m/s for each species; Antarctic Fur Seal (AFS), Macaronic Penguin (MP) and King Penguin (KP). Thick line in centre of box indicates the median, top and bottom of box is the upper and lower quartile and the range of the data is shown by the whiskers.*

Table 5.7. *ANOVA of percentage locations removed with a maximum speed of 3 m/s. Main effects are Filter and Species with the interaction effect indicated by the row, Filter:Species.*

	df	SS	MS	F	p
Filter	1	4724.7	4724.7	132.8222	< 2e-16
Species	2	182.7	91.4	2.5687	0.07823
Filter:Species	2	218.9	109.4	3.0764	0.04750
Residuals	317	11276.2	35.6		

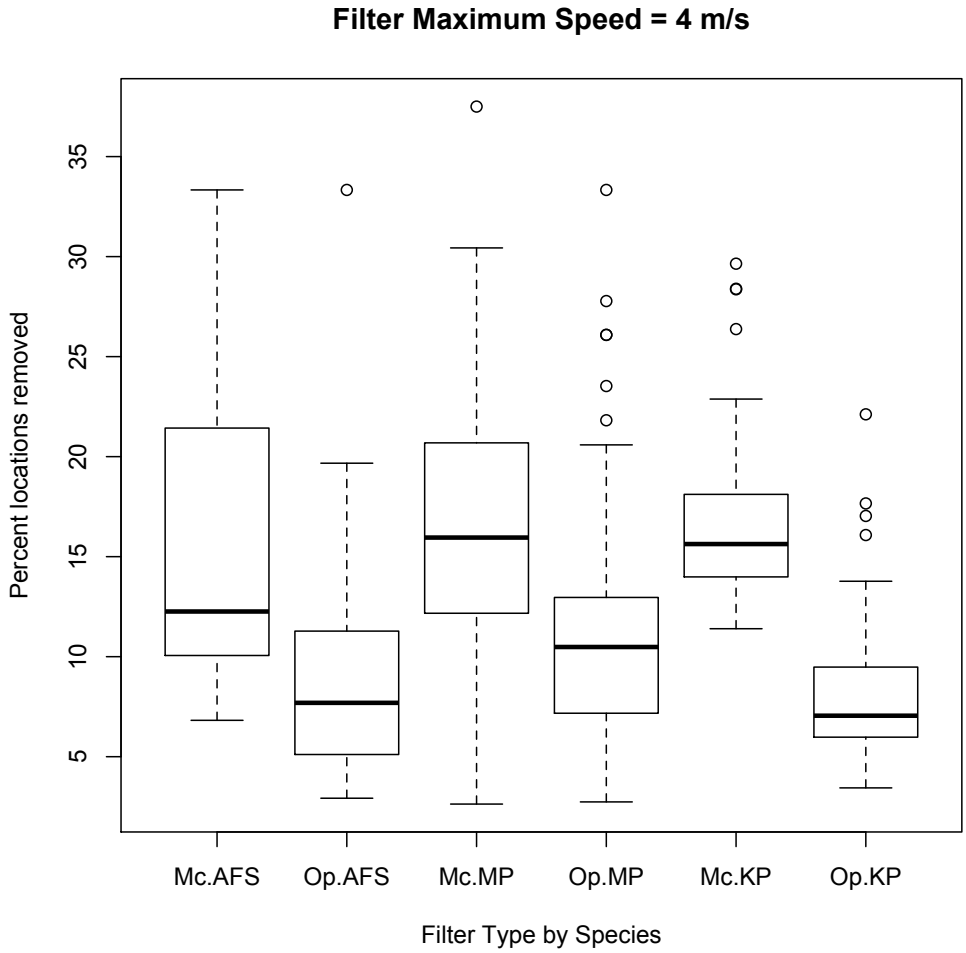

Figure 5.6. *Boxplot of percentage locations removed by McConnell filter (Mc) and Optimising filter (Op) at maximum speed of 4 m/s for each species; Antarctic Fur Seal (AFS), Macaronic Penguin (MP) and King Penguin (KP). Thick line in centre of box indicates the median, top and bottom of box is the upper and lower quartile and the range of the data is shown by the whiskers.*

Table 5.8. *ANOVA of percentage locations removed with a maximum speed of 4 m/s. Main effects are Filter and Species with the interaction effect indicated by the row, Filter:Species.*

	df	SS	MS	F	p
Filter	1	3192.3	3192.3	100.8720	< 2e-16
Species	2	192.9	96.4	3.0470	0.04889
Filter:Species	2	180.6	90.3	2.8527	0.05917
Residuals	317	10032.1	31.6		

Table 5.9. *ANOVA of percentage locations removed with a maximum speed of 4 m/s. Main effects are Filter and Species with the interaction effect indicated by the row, Filter:Species.*

	df	SS	MS	F	p
Filter	1	15279	15279	209.4092	< 2.2e-16
Species	2	2855	1427	19.5627	4.358e-09
Speed	3	209581	69860	957.4546	< 2.2e-16
Filter:Species	2	945	472	6.4725	0.0015998
Filter:Speed	3	1372	457	6.2664	0.0003212
Species:Speed	6	1766	294	4.0342	0.0005212
Filter:Species:Speed	6	198	33	0.4522	0.8437146
Residuals	317	87776	73		

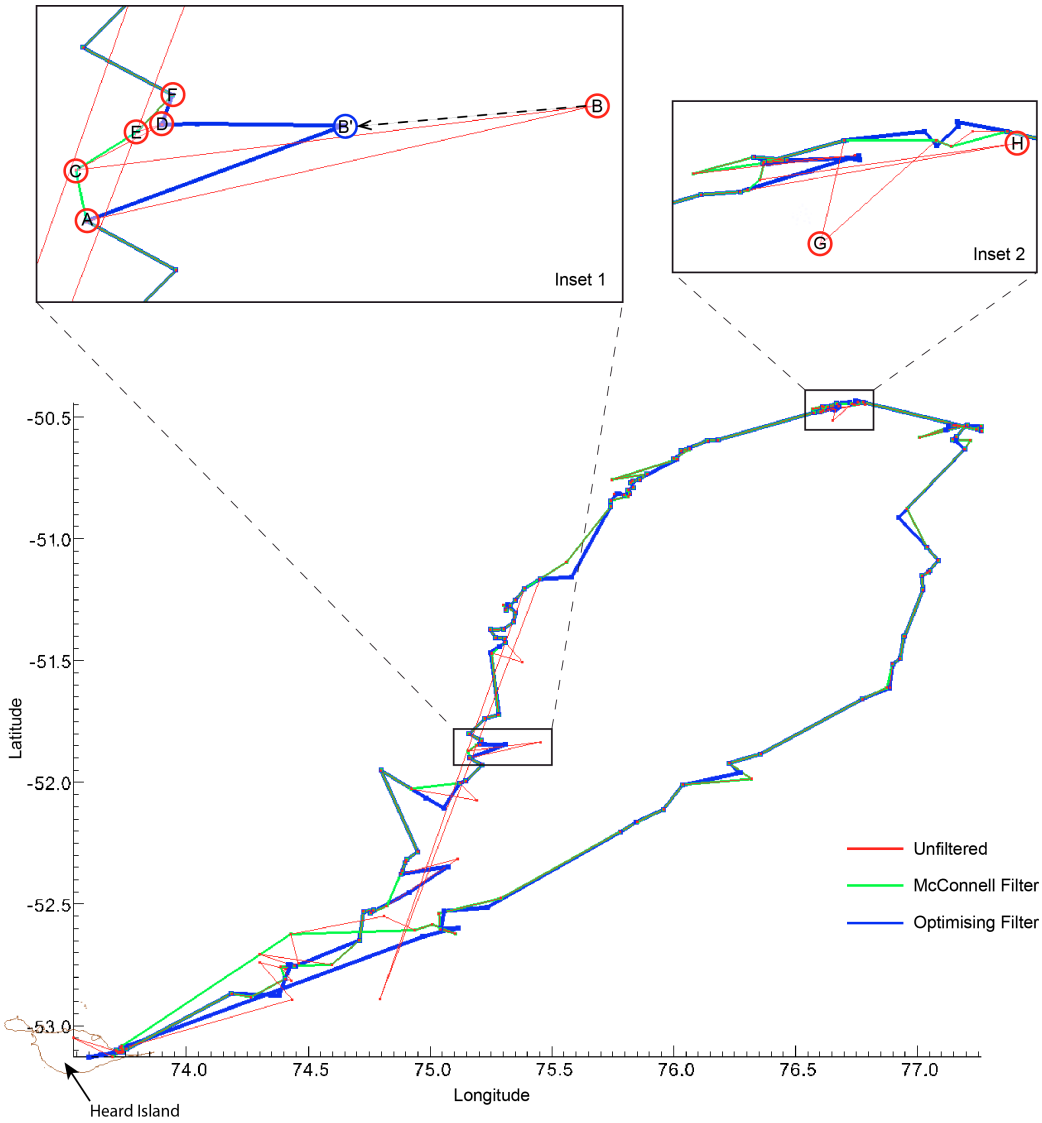

Figure 5.7. *Map of an Antarctic fur seal's foraging trip north-east of Heard Island. Unfiltered locations with interconnecting lines are shown in red. McConnell filtered locations are in green and locations produced by the optimising filter are shown in blue. Insets 1 and 2 show magnified views of sections of the track.*

An example track showing the effect of applying each filter at 3 m/s to locations from an Antarctic fur seal is given in Figure **5.7**. Inset 1 demonstrates how these filters work. The unfiltered locations in red show a track with the sequence of locations, A, B, C, D, E, F. The green McConnell filtered track shows how locations B and D are removed thereby creating a sequence of movement that passes directly through locations A, C, E and F. The optimising filter in blue does not remove the location B but rather adjusts its position to create location B′, thereby reducing the maximum speed travelled between locations A and B′ and locations B′ and D. In this case location C was removed,

giving a final set of locations, A, B′, D, E and F. Inset 2 shows how the two filters identify the same unfiltered locations, G and H for removal while generating similar results for most of the finer scale location data.

5.4 Discussion

Argos location data is usually supplied with error estimates, yet this information while often reported, is not always used quantitatively. In this study I have developed a filter that uses the concept of location regions to incorporate location error into its core algorithm. I have shown that even when given nothing more than a simple criteria such as maximum speed, it is possible to produce a filtered track with significantly less locations removed than is achieved through the use of the filter described in McConnell et al. (1992).

Many studies have sought to measure attributes of swimming speed in marine animals (Boyd et al. 1995; Hindell & Lea 1998; Wilson et al. 2002). It therefore makes sense to use this knowledge in designing a filter to identify erroneous locations in a data set that is known to represent a sample of an individual's movement over an extended period of time. When this kind of filter is applied to Argos location data, a problem can occur when locations are close together. This is due to the effect of the size of the location error relative to the spatial or temporal distance between locations (Hays et al. 2001; Vincent et al. 2002). When locations are close together, the location error has a greater effect on the measured speed and can therefore cause locations to be falsely rejected. Another way of stating this is that the closer two locations get to each other, the lower the signal to noise ratio and the less effective, simple point location based filters such as McConnell et al. (1992) are at correctly identifying erroneous locations. The algorithm described in this study solves this problem by incorporating potential location error into the filter and treating the Argos data as location regions rather than fixed points. This means that when locations are close together and significant error would cause a false high speed estimate and subsequent incorrect rejection, the optimising system can simply adjust the estimated location to keep the speed within limits. As long as this adjustment does not exceed the pre-set error boundaries as determined by location class, then the location will continue to be included. In addition to helping to retain more

locations, another potential advantage of this method is the greater fine scale detail that is given by a filter that removes fewer closely positioned locations.

One advantage of using a metaheuristic form of optimisation (Blum & Roli 2003) is the relative ease with which objective functions can be modified or added to the system. All that is needed to create an objective function is to provide a scalar assessment of the current system (in this case location positions) that allows the algorithm to grade a mutation's effect on the system relative to other mutations (Sean 2009). In the current study, only two relatively simple objectives were developed. The intention was specifically to create a more efficient form of location filtering than that offered by McConnell et al. (1992). This filter makes no assumptions about the data other than to impose a maximum speed and to incorporate empirically determined levels of location error. There is, however, no reason not to put more sophisticated constraints on the system. For instance, constant maximum speed could be replaced with a formula that varies the maximum speed based on other parameters such as energetic equations or diving behaviour in the form of diving records obtained with archival depth recorders (Naito et al. 1990; Arnould & Hindell 2001; Kuhn et al. 2006). Another possibility would be to modify the filter algorithm so that it does not discard locations. Instead, all locations, in particular the non-bounded LC 0, A and B would be modified until the track met all objective constraints. The output of the filter would then be a track with the same number of locations as the original but with some locations shifted potentially large distances from their original position. In the absence of more accurate data such as GPS fixes, this information could then be used as an estimate of the actual error size, calculated on a per location basis. Furthermore, the addition of GPS based tags (e.g. Costa et al. 2010; Patterson et al. 2010) could be used in such a study to both verify and fine tune the accuracy of this new filter.

According to Argos (2008), an elliptical representation of location error is more accurate than a circle. Although this information was not available at the time that the data in this study was collected, Argos now provide data on the size and orientation of this ellipse on a per location basis. In this chapter I have approximated the elliptical error through the use of empirically determined latitudinal and longitudinal location class error rates (Vincent et al. 2002). It is

expected that future versions of the optimising filter will be modified to incorporate the new per location error estimates into the design of the mutation operator.

When the optimising filter makes an adjustment to the position of the Argos location, it does so within the boundaries defined by the location's error estimate. Since the error estimate is determined in reference to the location supplied by Argos, it follows that the error estimate of the adjusted location may also need to be modified. Further research is required to determine an equation to create a new error estimate as a function of the location's error for latitude and longitude in conjunction with the size of the location adjustment. Such a study would require paired data sets of animal tracks that contain both the Argos estimated location and the actual location. It is only with the recent development of Fastloc technology (Costa et al. 2010) that obtaining such data has become feasible. As this data was not available at the time that the current study took place, it was therefore not possible to define a suitable equation for adjusting the error estimate. Consequently, the utility of the optimising filter technique as it is presented here is limited to studies whose methods are not dependant on a quantitative estimate of the location errors. Methods such as the model interpolated kernel smoothing algorithm presented in Chapter 4 which do use this error data are therefore unable to make use of this filter at the present time.

One obvious advantage of the McConnell et al. (1992) filter is its simplicity and ease of implementation. The optimisation filter presented here requires a relatively complex set of programming code. Also, its large volume of floating point calculations means that it is necessary to write this algorithm in a high performance expert language such as C or C++. The purpose of this chapter is to provide readers with a readily available alternative to the McConnell filter. To achieve this, a software platform was developed (see Chapter 6) and a free version of this system will soon be made available.

In summary, the optimising filter presented here provides an efficient filter for the detection and removal of erroneous locations in Argos animal movement data. It was shown to consistently outperform the McConnell filter when applied to different species and across a range of maximum speeds. Furthermore, by building on a generic optimisation framework, it offers the opportunity for

continued development and the potential for the creation of more sophisticated, data driven filtering techniques.

Chapter 6

Software Design and Implementation

Abstract

A detailed description of the design and implementation of a dedicated animal tracking analysis package was presented. This system was built using object-oriented programming principles with high-performance languages and data analysis libraries. The system addresses all aspects of required functionality for a typical animal tracking study. These include data retrieval, database storage, filtering, analysis and visualisation. The package includes a sophisticated generic optimisation platform which was used to implement the various analytical algorithms described in this thesis. The system was designed to be readily extendible and to be accessible through external analysis environments such as R.

6.1 Introduction

The analysis of movement of far ranging animals using satellite tracking requires a number of processing steps. These steps can be summarised as: 1) retrieval of the data from the Argos system; 2) extraction of the raw data and association with metadata such as the animal or track which the location originated from; 3) storage of the data; 4) statistical analysis which may include pre-processing for erroneous locations; 5) interpretation and visualisation. In all cases, a researcher's ability to effectively handle this data can be significantly aided through the use of appropriate software.

The development of software for this thesis has occurred over several years and involved the use of a number of programming languages. Broadly speaking, programming has been used in both the design phase and the implementation phase of the development. In the design phase, the emphasis was on the development and testing of ideas and algorithms. In the implementation phase, it was often necessary to re-write part or all of the design code with the intention of creating functional blocks of code that were robust, bug free and useable in a real

world application. When developing software that is intended to be used by others in a reliable and intuitive way, it is important that a number of design considerations be addressed. Issues such as performance, efficiency, scalability and long term utilisation can determine the success or failure of a software system (Standish 1994). It was with these goals in mind that a software system for the storage, analysis and visualisation of satellite tracking data was designed. In this chapter, I will describe the development of this system and how it was used to implement the algorithms defined in earlier chapters. Furthermore, this chapter will serve as a reference for developers with object oriented programming skills who intend to modify or extend the functionality of the system.

6.2 Programming and Design Considerations

The original system that I developed for tracking data analysis was the HeardMap program described in chapter 2. This program was written entirely in Java (Flanagan 2005). Java is a hybrid interpreted and compiled language that supports object oriented programming. This is achieved by compilation of source code down to Java byte code which is in turn executed by the Java Virtual Machine (JVM). The advantage of this approach is that the developer is effectively insulated from machine and platform specific details (Flanagan 2005). In theory, the same set of byte code should run on different versions of various operating systems without change. In practice there are often issues to do with user interface design, file handling etc. that complicate the cross platform ideal of this approach (Darwin 2001). Another potential downside to this language is that it often requires a lot of code to achieve tasks that other interpreted languages such as Python (Beazley 2006) and R (Venables & Smith 2009) can accomplish with relative ease. R is a free implementation of the proprietary language, S. It is an interpreted language which allows users to interact directly with the interpreter both to execute individual commands and to enter more extensive functions. It is designed expressly for the analysis, interpretation and visualisation of numerical data. As such, it is an extremely powerful environment for statistical and mathematical exploration. R is not however, a general purpose programming language. It does not provide or directly couple with any of the modern, mainstream Graphical User Interface (GUI) libraries. Also, it does not supply,

nor is there available, any form of Integrated Development Environment (IDE). Since R has become somewhat of a de facto standard for data analysis in the quantitative biology community, the interpolation and filtering functions are made available in R as function calls that interface to the main software system.

Like R, Python is an interpreted high level language that users can interact with directly through a command line interface. Python, however is not designed with any particular function in mind and like Java can be pre-compiled to an intermediate byte code level. It is a true general purpose programming language that can used for operating system scripting, procedural programming and the development of small to large scale object oriented systems (Pilgrim 2010). It interfaces to all the major GUI toolkits and has a range of IDEs available for development. On its own, it would not offer an alternative to R's analytical powers except for the availability of extensive libraries for scientific analysis. In particular, NumPy (Numeric Python) (Oliphant 2006) and SciPy (Scientific Python) (Eric et al. 2001). SciPy, is built on top of NumPy and when combined with graphing libraries such as MatPlotLib (Hunter 2007), offers similar levels of analysis functionality to systems such as R and Matlab (Moler 2004). Python was chosen for the design phase as it not only allowed for developing optimisation concepts but also to explore ways of creating an object oriented framework to house the filtering and interpolation algorithms

The use of Python for design work and initial programming enabled quick testing of prototype ideas and their functionality. However, Python's strength for design as a high level interpreted language is also its weakness in terms of performance (Juneau et al. 2010). Interpreted languages, even when precompiled to a byte code layer are never as fast as a natively executing program. This is why languages such as C and C++ are often used for performance intensive applications (Walters 2001). This is due to C being a language that compiles down to native machine specific executable code (Kernighan et al. 1978). C++ is a superset of the C language that compiles natively while adding object oriented constructs (Stroustrup 1997). All processor intensive tasks were therefore written in C and C++.

Evolutionary based optimisation algorithms work on the concept of exploring a given search space by repeatedly modifying existing solutions using a

mutation operator that imposes some form of random variation. By defining objective functions that quantitatively assess the value of each mutated form, various techniques of selection can be used to iteratively improve the overall fitness of the population and consequently converge on a solution. As in nature, many of these mutated forms will get rejected leaving only a small subset of the variation to find its way to the next generation. From a practical standpoint, this can lead to very large quantities of calculations and therefore high processor usage. This performance cost was managed by writing the optimising system entirely in C and C++.

Qt is a cross-platform application and user interface development framework. Qt allows the development of modern, professional and sophisticated graphical user interfaces (GUI) that work equally well on all major operating systems (Blanchette & Summerfield 2008). It is written in C++ and also has a set of Python bindings, making it ideal for this project. Additionally, it comes with a full suite of support classes separate to its GUI library that include various container classes for low level data storage and manipulation.

Often the utility and applicability of a programming language to a task is driven as much by its own features as the availability and cost of accessory code libraries. The Visualization Toolkit (VTK) is a large library of over 1000 classes that provide a complete solution to data visualization in 2-, 3- and 4-D (Schroeder et al. 1998). VTK is written in C++ but is also fully operable from languages such as Python and Java through the use of supplied interpreter bindings. As part of its architecture, VTK provides a pipeline system for data flows that connect functional analysis and visualisation modules thereby allowing the interpolation and integration functions to be easily linked in with existing VTK components such as contour plots, kernel smoothing and 3-D graphical windows.

While VTK does an impressive job of data flow management, analysis and visualisation, it is however simply a code library, not a functional application. In order to use VTK's functionality, it must be implemented within a GUI application. A number of VTK based data analysis applications have been developed and most of these are freely accessible in an open source format. The three major systems currently available are VisTrails (Callahan et al. 2006), VisIt (Ahern et al. 2000) and ParaView (Henderson et al. 2004). Of these, only

ParaView has the support of the software company, Kitware who are also the developers of VTK. ParaView offers a flexible, extensible and intuitive approach to data analysis. Furthermore, it can be configured to run in a client server arrangement such that large scale models requiring massive amounts of parallel processing even at the level of Cray supercomputers can be utilised (Patchett et al. 2009). In the Track system, ParaView was used to visualise the output of the data analysis components.

The Unified Modelling Language (UML) is a methodology for requirements analysis and object oriented software design (Fowler & Scott 2000). At its core, it consists of a set of diagrams that are used to define the structure and function of a software system. In this chapter, component diagrams and class diagrams will be used to describe the static structure of the system. The dynamic behaviour of the system components will be represented with activity diagrams.

6.3 Database Design

The Argos satellite system that provides the location fixes for animal tracking delivers its data as a text message using a format known as Diag (Argos 2008). These files must first be parsed and stored before the data can be used. During the development of the HeardMap program (chapter 2), a Java utility was developed that would both parse the email text and store the result in a database for later use. The system described in this chapter has similar functionality with a Qt based interface that allows users to quickly and easily manage large stores of animal tracking data as well as selecting tracks or groups of tracks for further analysis.

Animal movement data such as those provided by Argos is not in a form that is readily useable by researchers. The data must first be parsed and then stored in a database or file system that permits easy retrieval. Parsing of the raw data files was achieved with a C++ class that was written using the Argos file format (Argos 2008) as a guide. Once parsed, the raw data is stored in a relational database that collates the data into sets of locations that relate to the track of an individual animal. This track metadata record also stores the GPS location that the tag was deployed at and optionally the location where it was retrieved. Also notes, e.g. problems with deployment, can also be recorded here.

SQLite is a database written in C and is optimised for both speed and size. SQLite stores the entire database structure and contents in a single file and is industry proven to be robust and reliable (Owens & Owens 2006). The schema diagram (Figure **6.1**) shows the structure of the database tables and their primary to foreign key constraints. All tables in this database have an auto-incrementing integer primary key as their first column.

Argos makes a raw unprocessed data stream available to its subscribers which it calls a Diag format(Argos 2008). This data is loaded without modification into the table, ArgosDiag (Figure **6.1**). The column names in this table match those of the Diag data definition. The structure of this database enables users to access the data either as individual tracks or as groups of tracks. Locations within a track are found using their unique platform transmitter number and the beginning and end dates of the transmitter's deployment. By using this information which is stored in the ArgosTracks table, a SQL Select query is able to extract a subset of locations stored in the ArgosDiag table. Often it is necessary to extract related groups of tracks as a single data stream, e.g. when analysing the area usage of a cohort of seals. In this case, the ArgosTrackGroups and ArgosTrackGroupLink tables can be used in a second SQL query to create groups of location tracks.

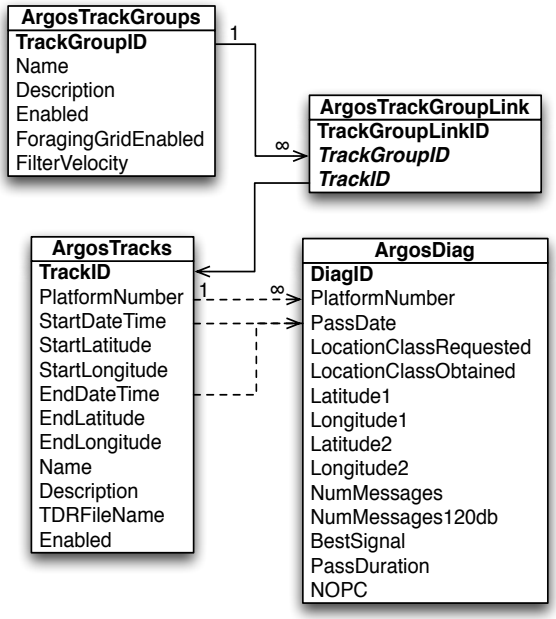

Figure 6.1. *Database schema diagram showing the field structure of the tables. Solid lines indicate key constraints while dashed lines show the implied logical relationships. Keys are in bold font with foreign keys italicised. The direction of one-to-many relationships are added to lines by the standard convention of a 1 and ∞ symbols.*

6.4 Component Overview

It is expected that this system will be used in a number of different ways, for instance, as part of a larger analysis within R or incorporated as a component in a related software application. It was therefore necessary to make component re-use an overriding goal throughout all aspects of the system design. This was achieved through the use of object oriented design principles such as inheritance and polymorphism (Stroustrup 1997). Source code is organised in a hierarchical fashion where functional system components (see Figure **6.2**) incorporate classes that, wherever possible, take advantage of class inheritance.

In the component system collectively known as Track (Figure **6.2**), the Optimizer contains the core processing algorithm as defined in chapters 3, 4 and 5. This component contains a set of classes that work closely together to form a generic evolutionary optimization system. This component is designed to be highly configurable so that it can be utilised for a variety of tasks.

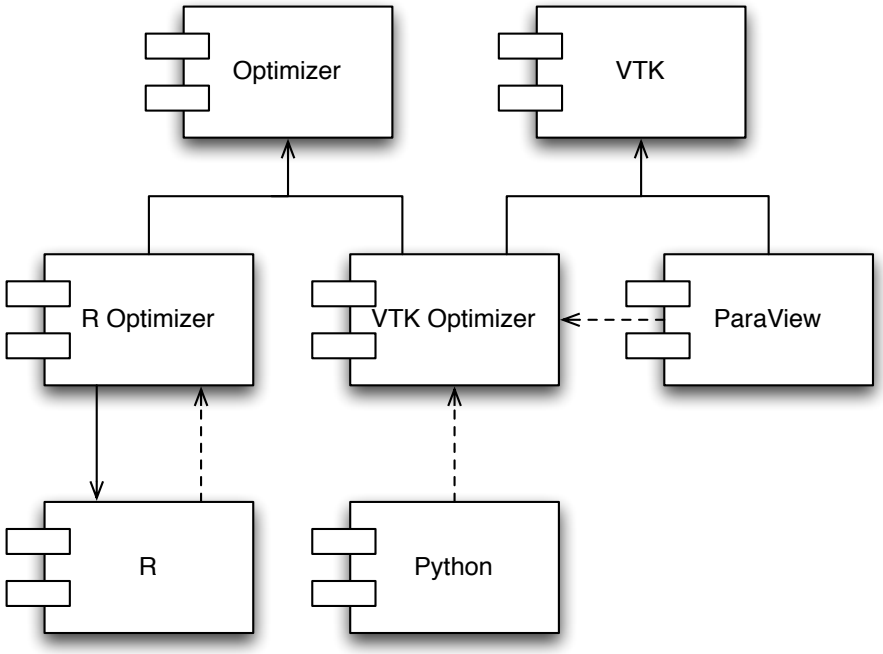

Figure 6.2. *Component Diagram of the Track software system. Where a component's functionality or coding is dependent on its link with another component, this relationship is indicated by a solid line which represents a UML association. If the relationship can be optional such as choosing whether or not to incorporate this system into an existing R installation, the connection is represented by a dashed line that is the convention for a UML dependency. The arrows indicate the direction of the association or dependency,* e.g. *the R Optimizer component cannot be built without connections to the Optimizer component though the Optimizer has no need for or knowledge of the R Optimizer.*

The VTK component refers collectively to the VTK source code library. The classes contained in this library are intended to be used through object oriented inheritance while the library itself should remain unchanged. The VTK Optimizer utilizes the VTK library component classes to assemble the Optimizer classes into working solutions. It is at this level that systems such as location interpolation (chapter 4) and speed filtering (chapter 5) are constructed. The R Optimizer is analogous in function to the VTK Optimizer except that it uses R based parent classes and mechanisms to establish input and output data flows.

The R, Python and ParaView components are conceptual links to their respective applications. In all three cases, these applications are designed to be readily extendible through the addition of external modules. Like VTK, the source code for the core applications should not be modified.

6.5 Optimizer Structure

The Optimizer class (Figure **6.3**) forms the central controller for a set of concrete and abstract classes that work together to provide a multi-objective optimization framework. The class is firstly, a container for data in the form of a LocationTrack object. In Figure **6.3**, the LocationTrack object is represented as a package because this data store is built from a set of classes that will be described below. The optimizer also contains one or more objectives and any number of observers. In this context, an optimizer objective refers to the mathematical function that the system is attempting to minimise while an observer is any class that is notified of changes in the system state so that that information can be stored for later use. These objects are all stored in instances of the Qt container class, QList which provides a generic linked list implementation (Blanchette & Summerfield 2008).

Once the state of an instantiated Optimizer has been established, the optimizer is then controlled via calls to the init() and iterate() methods. The init() method must be called at the start of the optimisation process as it initialises the state of all internal variables. The iterate() method is then called once for each iteration of the optimisation. This class also has a number of utility methods that are used within the implementation of init() and iterate(). These methods are shown in Figure **6.3** with a '-' prefix to indicate their scope as private. These methods are not intended to be used outside of the optimiser.

The Pareto dominance ranking approach to multi-objective assessment (see chapter 4), is implemented within the optimizer through the use of a QList of the data structure called Rank. The ranking of each mutation fitness is stored by the order of objects in this list after a call to sortRankedMutations(). The selected mutation is chosen by calling random_i() which returns a number from a random power law distribution.

Two abstract classes, Objective and Observer are also supplied. Apart from maintaining a pointer to the optimizer instance, these classes define the contract which subclasses must implement. The objective forms an integral part of the workings of the optimization and it is therefore necessary for at least one objective to be present for any form of optimization of the LocationTrack data to

take place. When designing an objective subclass, it is important to note that there are two different fitness methods. The first, fitness() is expected to return a value that is an absolute assessment of the current set of locations. This method is called once per iteration. The mutationFitness() method however, only needs to provide a number that indicates the relative change to fitness that the given location mutation would cause if it was selected. It is important to keep calculations in this method to a minimum as this method is called once per location during each iteration of the optimisation. The idea for an Observer class comes from the design pattern of the same name (Gamma et al. 1995). The presence or absence of observers has no bearing on the functioning of the optimizer. They are however important as this is the primary mechanism for storing the results of the optimization. Without at least one observer in play, the only information available would be the current state of the system as past state has no function in the workings of the optimization algorithm. Two Observer implementations are supplied. The TrackLinkRecorder produces a historical record of changes to the links between locations while the TrackPointRecorder does the same for changed locations.

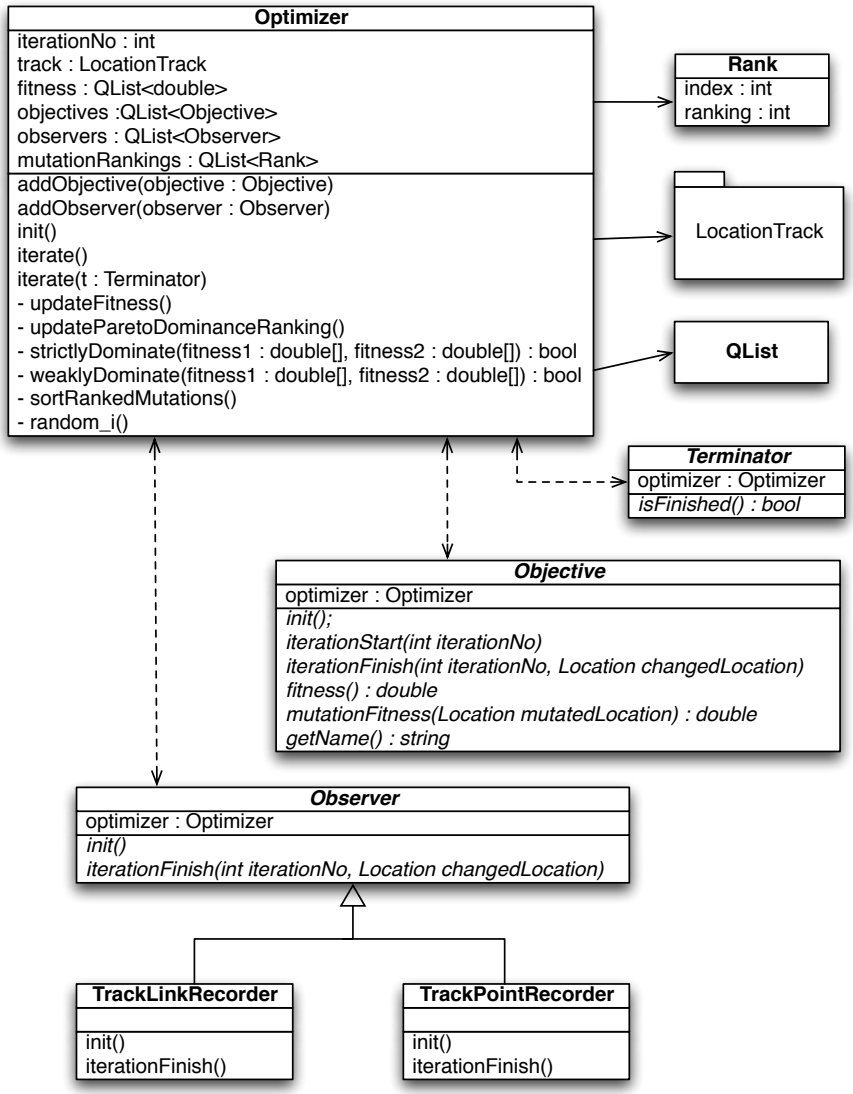

Figure 6.3. *Class diagram of the core optimizer algorithm. The Optimizer class and its associated classes are shown with their relationships drawn between them. Class inheritance is indicated by the triangle arrow head. Associations are shown with a solid arrowed line and dependencies with a dashed arrowed line. Abstract classes have an italicised name as do their non-implemented methods.*

The LocationTrack package (Figure **6.4**) has three core classes, LocationTrack, Location and Link. Also included are three application specific classes that implement the abstract class, Location. These are ArgosLocation, InterpolatedLocation and TerminalLocation. The LocationTrack class utilises a QList to store a list of objects with a Location super class. In addition to some administrative methods for maintaining the contents of the locations list, this class also has two helper methods that permit the optimizer to initiate location mutations. These are mutateLocation(Location) which forces a random mutation of the specified location and mutateLocations() which does the same for all

locations (see chapters 4 & 5). These methods serve to encapsulate and hide the internal workings of the LocationTrack package as well as managing the update of the mutation fitness vectors that are utilised within each iteration cycle.

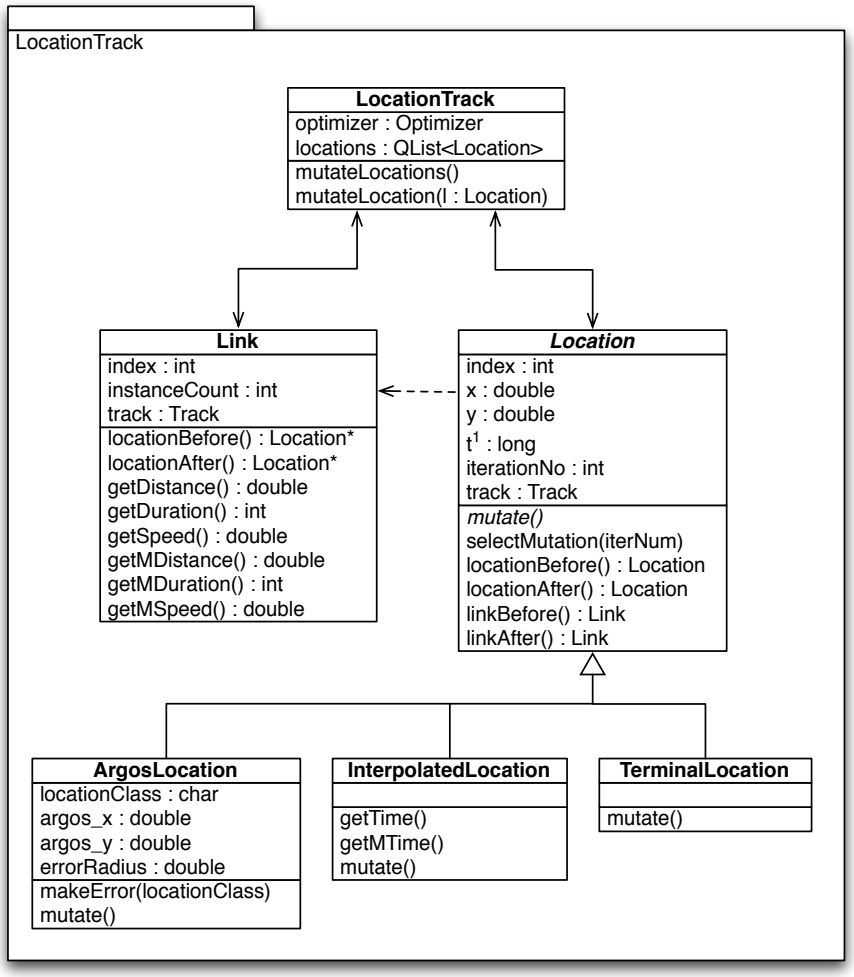

Figure 6.4. *Class diagram of the LocationTrack package. This package is a logical representation of the LocationTrack class and its associated classes.*

The design for a set of locations represented as a track was driven by the dual perspective nature of the optimizing algorithm. In this system, the mutation operator works on specific locations within the track. The objective fitness assessment however, has the option of viewing the track as a set of links that join these locations together (Figure **6.5**). See Chapter 4 for the rationale behind this.

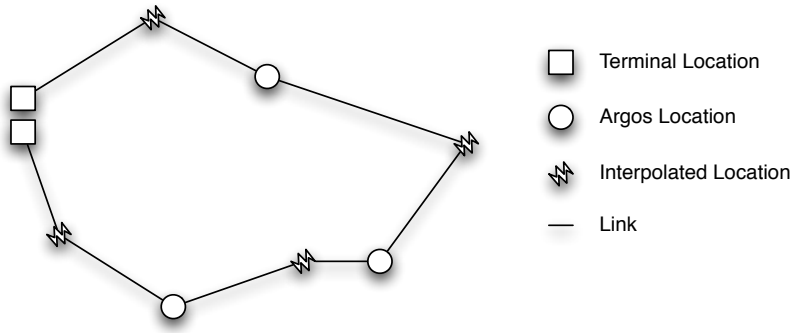

Figure 6.5. *Simple example of how an animal track is represented and stored within the LocationTrack package.*

The three types of locations Terminal, Argos and Interpolated are polymorphic forms of their parent Location class. The Location class stores basic spatial and temporal attributes in the form of x and y coordinates and time using the standard number of seconds since midnight on 1 January 1970 UTC (Kernighan et al. 1978). This class also provides methods for navigating neighbouring Links and Locations as well managing the update of data when an iteration of the optimizer decides on a mutated location to select.

Terminal locations, when used, are the first and/or last locations in a track and by default are assumed to be exact positions, i.e. a known location of deployment and retrieval. Their use is optional as this information is not always available. The behaviour of this class is to ignore all calls to mutate, it always returns the same value for x, y or t and does nothing in response to the mutate() method.

The ArgosLocation class maintains the initial latitude and longitude values as a record of the actual location reported by the Argos data set. This class also stores a location class (LC) attribute which is the Argos system's name for estimated error (Argos 2008) and it has a method for modifying the LC into a quantitative error distance in metres. This method defaults to a lookup table of values reported by Argos but can easily be modified to a different set of values such as those that may be obtained experimentally (e.g. Vincent et al. 2002). The mutate() method which facilitates the algorithm's ability to search is overloaded to implement the mutation operator defined in chapter 4.

The InterpolatedLocation class stores its current position with the x and y properties but does not store time as ArgosLocation does. This location returns

time as a function of its relative position between its adjacent ArgosLocations (see chapter 4).

The Link class returns properties that describe its linear connection between two locations. It has methods to calculate its distance, duration and speed but maintains no long term state. By conforming to the Flyweight design pattern (Gamma et al. 1995), this class reduces memory use by calculating property values on request and using the values stored in the adjacent locations. Consequently, there only needs to be one instance of this class per track. This is purely a design choice and of no concern to the user as the illusion of *n-1* links with *n* being the number of locations is still maintained. This class also overloads mutate() and implements the mutation operator defined in chapter 4.

6.6 Optimizer Operation

The optimizer produces solutions by calls to the iterate() method (Figure **6.7**). Before calling this method, it is first necessary to initialise its operation through a single call to init() (Figure **6.6**). Keeping the initialisation code separate from the class constructor permits reuse of the same optimizer object for different tracks. The init() method performs administrative tasks such as creating internal array stores. It then notifies the objectives and observers in order to give them a chance to make their own preparations and finally it calculates the initial fitness state of the system by calling the objectives.

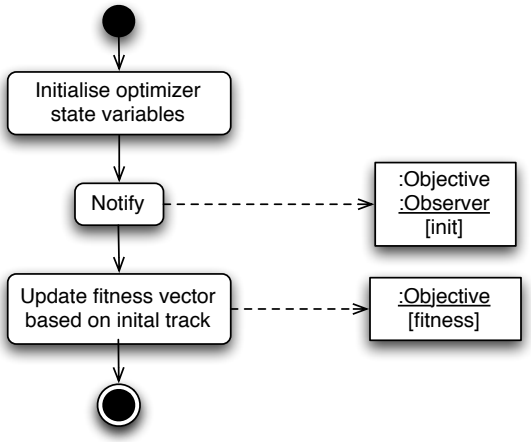

Figure 6.6. *Activity diagram of the sequence of events that occur when init() method of the Optimiser class is called. The solid circle represents the start while the double circle indicates the end. Rounded rectangles are activities and solid arrows indicate the sequence. The dashed arrows show the flow of messages while the rectangles contain the message receiving classes above the line and the method being called below the line.*

Repeated calls to iterate() produce continuous modifications to the track by a process of objective guided mutation selection. This is the method that implements the algorithm described in chapter 4. The iterate() method has two forms. The no argument version produces a single shot iteration. The overloaded version allows for multiple iterations. The typical use case will be to provide a Terminator subclass to iterate() that implements the isFinished() method. Such a class might simply count the number of iterations completed or in a more sophisticated design, could examine the quality of the solution before deciding to stop. When called, iterate() first notifies the objectives and the observers that the process is about to start by sending the iterateStart() message. Each location is then mutated and the effect on fitness is measured by calls to the mutationFitness() method of each objective. The Pareto dominance ranking of each location is then determined by first counting the number of locations that strictly dominate a given location and then sorting the indexing array by that count. Using a function that returns a random number from a power law distribution, a mutated location is selected. This selection is then committed to the track by calling selectMutation() on the location object. The fitness of the new track is updated by calling the objective's fitness() method and finally, the objectives and observers are notified that the iteration has finished.

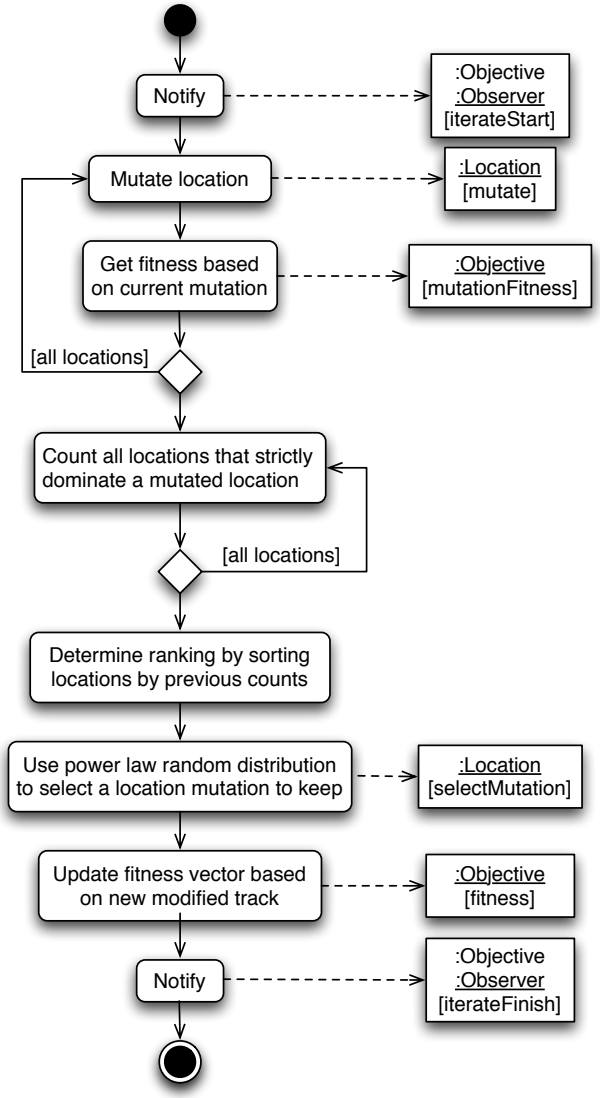

Figure 6.7. *Activity diagram of the sequence of events that occur when iterate() method of the Optimizer class is called.*

6.7 Optimiser Application

Figure **6.8** demonstrates the general approach to incorporating the Optimiser system within a component of a larger system such as the R Optimiser or the VTK Optimiser. By assembling various combinations of objective, observer and terminator objects, it becomes possible to configure different optimization functions. Two examples of the application of the optimizer are speed filtering (chapter 4) and location interpolation (chapter 5).

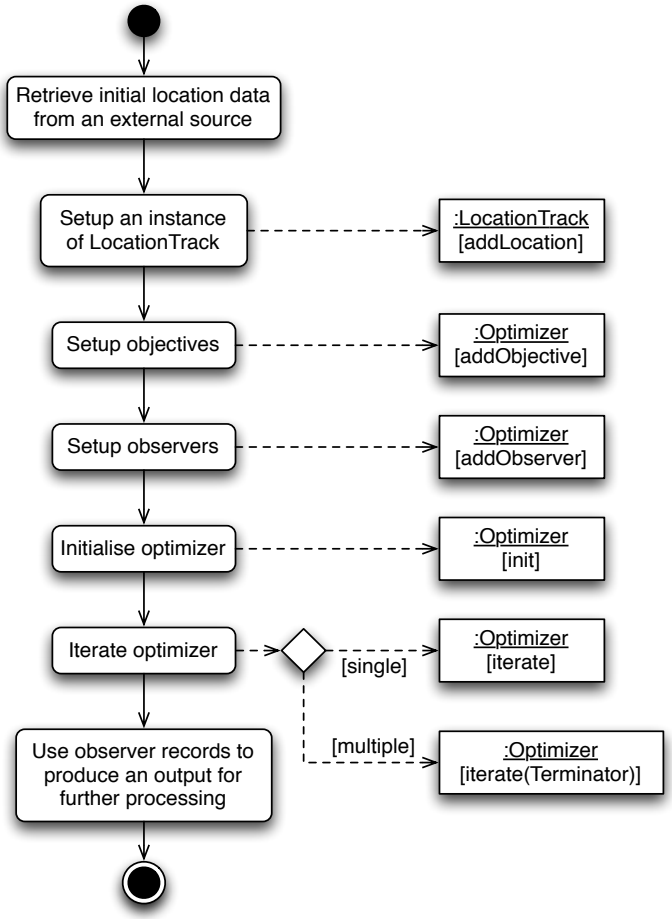

Figure 6.8. *Activity diagram of the suggested sequence of steps required to create an optimization solution.*

The first step in building an optimizing solution is to setup the LocationTrack object with an initial set of locations. This will most likely come from an external source such as a database or application data stream such as the VTK pipeline. Objectives and observers are then added to the optimizer object and then the init() method is called. Iteration is then started either by calling the no argument single shot method or by supplying a Terminator object to allow automatic looping from within the body of the optimizer class. Once optimization is complete, the results can be compiled from the observers and output to the next stage of analysis.

6.7 Interfacing the Optimizer to External Environments

The three environments that have been chosen to utilise this optimizer based location analysis system are R, Python and ParaView (Figure **6.2**). Since the core system is written in C++, the task of interfacing to R (R 2008) and Python (Paul

2007) is relatively straightforward. In both cases, it is simply a matter of writing wrapper functions that call the optimizer and after recompilation, the full functionality of the optimizer will be available within these environments.

To build the system into ParaView first requires the intermediate step of creating VTK algorithm subclasses. In the two applications of the optimizer that have been developed for this study, the VTK subclass is used to contain all the functionality that produces both the speed filter (chapter 5) and the location interpolator (chapter 4). Once the VTK class is created, incorporation into ParaView is achieved by the creation of two XML files and some minor changes to the VTK header file (see Henderson et al. (2004)).

6.8 Conclusion

This chapter has given a detailed description of the design and development of software that produces an evolutionary optimization based approach to animal tracking data analysis. This software system (Track), is intended to provide an intuitive and robust solution that is readily accessible to researchers catering for varying degrees of knowledge, experience and technical ability. A design that permits multiple entry points to accessing the optimization algorithm was created to achieve this. This design allows access to the optimizer as a programming library, an R or Python based plug-in or through a customised version of ParaView, it is hoped that this system will not only provide analysis and visualisation solutions but also promote some methodological standardisation amongst the global community of bio-logging researchers.

Chapter 7

General Discussion

7.1 Introduction

A researcher's ability to ask spatially motivated, ecological and behavioural questions is to a large degree, dependant on the quality of the data and its subsequent analysis. When these data are to be obtained from geographically remote, far ranging animals such as seals or penguins in the Southern ocean, telemetry based technologies are required. Two commonly used approaches for the acquisition of geographical location data are 1) light level geo-location (Phillips et al. 2004) that roughly estimates position based on the time of onset of sunrise and sunset and 2) the Argos satellite system (Argos 2008) which provides a complete service for remote location tracking through the use of small, lightweight animal-borne transmitters. More recently, a new, more accurate, global positioning system (GPS) satellite tag called Fastloc has also become available. Each of these technologies has various advantages and disadvantages in terms of accuracy, cost, ease of use and reliability. The data set used in this thesis was obtained from an expedition to Heard Island where numerous Argos platform terminal transmitters (PTT) were deployed on individual animals over a forty day period. The algorithms presented have been developed using this data set and therefore deal directly with Argos data. However, much of this work could with some modification be applied to other sources of animal movement data.

This study has explored the development of various quantitative tools for the analysis of animal track data as supplied by Argos. In this chapter, I will discuss the outcomes of this thesis and present ideas for future directions that may occur in response to this work.

7.2 Predicting the Unknown

Numerous methods have been developed over the years to address the issue of location uncertainty. The simplest approach is to discard some or all of the location classes that do not have an estimate of error, i.e. 0, A and B (Cardona et al. 2005; Pinaud & Weimerskirch 2005). More sophisticated techniques such as state space modelling that allow for the integration of multiple probability distributions in the prediction of future movement have also been suggested (Jonsen et al. 2003; Tremblay et al. 2009). The most commonly used approach however, is the application of a simple speed filter (McConnell et al. 1992). This filter works to remove locations that would be deemed improbable due to the speed required for the animal to traverse to that position. A problem that can arise from the use of this kind of destructive filter is the false detection and removal of potentially valid data (Freitas et al. 2008). This can occur as a result of the application of incorrect or overly simplistic assumptions in the decision criteria for data removal, e.g. if maximum speed is set too low then faster but legitimate locations will be lost. Difficulties in identifying valid data are also caused by observational error. Location error is the product of several interacting factors including the technical limitations of the tracking system, environmental forces and the animal's behaviour (Costa et al. 2010). Location error can have the effect of distorting quantitative measures such as the determination of travel speed (Hays et al. 2001) as well as masking the correct description of location or the detection of behavioural states (Costa et al. 2010). The presence of location error is an unavoidable consequence of using the Argos satellite system for tracking marine animals. However, knowledge of the error in the form of a quantitative estimate, e.g. location class or location error ellipses (Argos 2008), can be incorporated into analytical techniques in order to make better use of the underlying location information. Techniques such as state space modelling (SSM) (Patterson et al. 2010) build a model of observational error into their calculations as part of the process of predicting the animal's future movements. While studies such as Patterson et al. (2010), pre-process the data with a destructive speed filter, SSM methods such as that described in Jonsen et al. (2005) manage to avoid the need for destructive filtering. In Chapter 5, a modified version of the extremal optimisation framework (Chapter 4) was used to

create a new technique for speed filtering of Argos location data. This filter, through the incorporation of location error estimates was able to produce a set of locations that meet the maximum speed criteria while requiring the removal of approximately half as many locations as that of McConnell et al. (1992). This method takes advantage of the error component at each location by permitting the adjustment of the Argos location coordinates within the boundaries of the estimated elliptical error region. It is not currently known whether these adjustments produce more accurate locations since the required validation data (e.g. GPS) was not available at the time. It is however quite possible that the optimising filter does indeed improve the accuracy of the Argos location data set.

The relative influence of location error and positional uncertainty between locations is a sliding scale that is dependant on the size of the temporal gap in the data. Using GPS location data obtained from an adult male grey seal (*Halichoerus grypus*), Lonergan et al. (2009) showed that when the gap between locations exceeds 12 hours, the effect of Argos location error on interpolation accuracy is negligible. However, when this gap is small, location error can have a significant impact on subsequent calculations (Hays et al. 2001).

Aside from the difficulties posed by the uncertainty around each location fix is the question of what the animal did in the period between location fixes. Infrequent and irregular location samples is a common problem with animal tracking data (Rooney et al. 1998). When large temporal gaps occur in the data, statistical methods can be used to compensate for this lack of information. Interpolation of these data allows for estimation of intermediate locations as well as providing the opportunity to produce a regularised sample data set (Lunardi 2009). Various options exist for interpolation of animal movement data. The simplest method is linear whereby each location is connected by a straight line. In treating the data this way, the researcher is making a definitive statement about the transitory behaviour of the animal. While, for some sections of the track this form of path reconstruction might in fact coincide with the animals actual movement, this technique does not allow for the possibility that it did not (Chapter 4). A more sophisticated approach, proposed by Tremblay et al. (2006), fitted a curvilinear track to Argos locations. The effectiveness of this technique was found to be species dependant with a higher degree of accuracy for

albatrosses than elephant seals. For many species, aside from the effect of ocean currents and wind, behaviour is the predominant factor influencing movement of an animal between known locations (Lonergan et al. 2009). It is therefore not surprising that the average movement patterns of some species (e.g. albatross) will be closer by chance to curvilinear shaped paths than others (e.g. elephant seals). Also, when total curvilinear track length was compared to that from linear interpolation, its estimate while still less than reality was closer than the linear result. Given that linear interpolation is the most conservative path reconstruction method and effectively defines the shortest distance between locations, any deviation from linearity will result in an increased overall length. Considering that it is unlikely that the animal only ever travels the unknown sections of a track in a straight line, it can be expected that the actual track length will always exceed that of the linear path. These findings demonstrate that by simply applying a Bézier or spline curve, immediate improvements in location estimation over that of linear interpolation can be achieved. Since animal behaviour, environmental influences and equipment functionality are all variable, it follows that methods that attempt to adjust their interpolation mechanism in response to these factors might provide a more accurate approach to location interpolation. Studies that incorporate models of behaviour and observational error into an SSM framework have shown this to be possible (Jonsen et al. 2003; Patterson et al. 2008). Similarly, the use of behaviour as measured by inter-location speeds for multiple individuals was used in this thesis to develop a model driven interpolation technique (Chapter 4). This method called model interpolated kernel smoothing (MIKS) also incorporates elliptical location error estimates. When compared to linear interpolation, it was shown to provide more accurate and consistent estimates of location.

The use of kernel smoothing for the estimation of spatial usage has been shown to be an effective technique when applied to animal movement data (Worton 1995; Blundell et al. 2001; Matthiopoulos 2003; Vokoun 2003). When applied to a small set of locations spread across a large spatial extent, a form of interpolation is achieved. This is due to the influence of each data point being spread across the resulting usage surface at a level that is determined by the kernel shape and size (Simonoff 1996). Reliance on kernel smoothing for interpolation,

even after weighting the data points (Chapter 3) therefore assumes that the choice of kernel is appropriate given considerations such as animal behaviour, location error and inter-location uncertainty. Given that this assumption may not be correct, I developed a technique for using models of behaviour and location error to drive an interpolation process that produces many possible scenarios of paths an animal may have taken (Chapter 4). In doing so, the intention was to provide a data set for the generation of spatial usage using kernel smoothing where the interpolation of the original location data was determined primarily by the model rather than the kernel. Furthermore, the overlap of these paths were then expected to converge or diverge in response to the level of uncertainty at a given spatial region. In this scenario, the primary function of the kernel was to act as a low pass filter (Simonoff 1996) that generates an output that is comparable to other animal spatial usage techniques.

7.3 Analysis Software

Two different software solutions were presented here. In chapter 2, the HeardMap program was developed with a specific study question in mind. This program demonstrated how an integrated software solution can provide substantial benefit to the success of a large scale ecological study. Many ideas and some functionality from this system was then migrated across to the more ambitious project, Track. In chapter 6, the library of software components called Track is explained. This system was designed to deliver much of the core functionality required by bio-logging researchers who use Argos satellite tracking data. The flexibility and utility of this system is demonstrated by its use in the various algorithms detailed in chapters 3 – 5. My intention with this system is to make it freely available through a web site in the near future. Additionally, I would expect that over time, the functionality of this system will be extended to incorporate other existing analysis techniques as well as further developments of the MIKS technique.

7.4 Future Directions

The algorithms and analysis presented in this thesis have provided working solutions to a number of issues that arise from the analysis of Argos satellite

location data. This work was undertaken with the intention of making a contribution to the animal tracking and spatial analysis literature. While an extensive body of work already exists in these fields, it is still a very active area of research. In this section I will describe some issues that have arisen in the development of this work and ideas for future improvements and ongoing research.

Spatial utilisation distribution (UD) maps were created using various forms of kernel density estimation (KDE) both with and without location interpolation. Validation of each of these techniques was achieved using multiple comparisons of randomised location track subsets (Chapter 3). Statistically, the results of these tests (Chapters 3 and 4), provided convincing evidence in favour of the use of model interpolated kernel smoothing (MIKS) over other simpler techniques. The use of MIKS also produced unexpected deviations away from a more intuitively expected linear travel path. Regardless of the strength of the statistically significant result in this analysis, the fact remains that the only true test of the applicability and accuracy of these methods is with knowledge of the actual movements of the animal in question (Matthiopoulos 2003). The addition of global positioning system (GPS) technology to the Argos satellite tracking system is now possible through the advent of Fastloc GPS (Rutz & Hays 2009). Recent studies have used this system to assess the error levels of the Argos location classes (Costa et al. 2010; Patterson et al. 2010). Similarly, this technology could be used to give valuable insight into the effectiveness of interpolation and smoothing. Such a study could provide definitive data on the ability to predict movement from sparse location data sets using techniques such as MIKS (Chapter 4) and state space modelling (Patterson et al. 2008).

The MIKS algorithm (Chapter 4) and its associated kernel smoothing methods (Chapter 3) all make use of the McConnell et al. (1992) speed filter to pre-process the data. The reason that the more efficient optimisation method for speed filtering (Chapter 5) was not used was due to the adjustments that this technique makes to the retained Argos locations. In shifting the actual location of the data, the distribution of error as defined by the location class is assumed to have also changed. In its current form, the use of this filter is restricted to studies that do not require quantitative estimates of location error. In a future study to

further develop this technique, a methodology to redefine location error estimates will need to be conducted. It is expected that such a study will require data from the recently developed Fastloc GPS tags. Once this is done, incorporation of the optimising speed filter into methodologies that utilise the location error distribution will become possible.

The concept behind the development of MIKS was to provide a configurable feedback mechanism that would guide the operation of an interpolation algorithm. The implementation of MIKS in Chapter 4 utilised a probability distribution function (PDF) of inter-location travel speeds as a measure of population behaviour. A goodness-of-fit statistic was used to assess the ongoing state of the interpolated data against the speed PDF. This process produced an interpolated UD with a structure that closely matched the measured population behaviour. In future applications of the MIKS method it is expected that other measures of behaviour may be incorporated. For instance, a common practice when tracking marine diving predators is to tag the same individual with an archival time depth recorder (TDR) (Boveng et al. 1996; Burns & Castellini 1998). The TDR is a small, battery powered instrument that uses a pressure sensor to store regular measurements of depth on a computerised memory chip (Burns & Castellini 1998). These tags are used to provide high temporal and spatial resolution data of an animal's vertical displacement in the water column (Chapter 2). When Argos location data and TDR dive data are collected in unison, the potential exists for integration of the two data sets in order to gain a three-dimensional perspective of the animal's movements (Georges et al. 2000; Hays 2004). Using MIKS, it would be possible to add an objective that restricts the surface level probability distribution based on an index of diving activity. The rationale for this approach being that if an animal is spending its time continuously diving, it would most likely remain in the same general location during this period. Of course, the validity and strength of this assumption would be species dependant so the design of this objective would include an adjustable parameter that could be set using knowledge of the species diving behaviour. An additional outcome could occur as a result of having more refined location estimates. The geo-referencing of diving activity which is achieved by synchronising acquisition times of the two data sets would therefore also be more

accurately defined. In Chapter 2, some preliminary location, dive integration functionality was described. Owing to the complexity of the location analysis that was presented in subsequent chapters, a thorough exploration of the integration of diving and location data was determined to be beyond the scope of this thesis. However, the existing data set from the Heard Island expedition that was used throughout this study does have a complete set of paired Argos locations and TDR dive records. It is therefore envisaged that the development of a technique for the incorporation of diving activity into the MIKS algorithm will be undertaken in the near future.

The aims of this thesis were necessarily focussed on the application of analysis solutions to Argos supplied data. However, the issues encountered when working with an inaccurate and irregular data source are not limited to satellite location tracking. The analysis techniques presented in this study all come down to the measurement, interpretation and manipulation of probability distributions. Given the generic nature of this statement, it follows that methodologies like MIKS may be applicable to many areas of research. Some possibilities to consider might be; light level geo-location of animal-movement that uses recorded twilight times to estimate location (Sumner et al. 2009); inclusion of other contributing factors such as weather, bathymetry and ocean currents; geographic layers that use knowledge of land masses or other potential obstructions to guide the system away from unfeasible locations e.g. a seal does not travel across land; non-animal related work such as modelling vehicular traffic flows for the purpose of reducing traffic jams (Karaaslan et al. 2009) or accident prevention (Tiwari 2000).

7.5 Conclusion

During the course of this study, I have addressed a number of issues that arise when attempting to develop a spatial model of animal movement from a limited supply of real-world data. As far as possible I have tried to develop data driven methodologies that make as few assumptions as possible. Additionally, I have worked towards delivering results that quantitatively acknowledge the level of uncertainty. This level of uncertainty is in turn directly attributable to the quantity and quality of the input data source or sources. In closing, this thesis

offers a number of applied solutions to the analysis of Argos derived animal tracking data. In using evolutionary based computation techniques, the methods presented here suggest a new approach to analysing these data that will hopefully inspire further research.

References

Ahern S., Bonnell K., Brugger E. & Childs… H. 2000. VisIt: a component based parallel visualization package. *Conference: Nuclear Explosives Code Developers Conference, Oakland, CA, Oct 23 - Oct 27, 2000*

Ajay M. 2010. Programming In C: A Practical Approach. *books.google.com*

Amstrup S., Mcdonald T. & Durner G. 2004. Using satellite radiotelemetry data to delineate and manage wildlife populations. *Wildlife Society Bulletin* 32: 661-679.

Anderson P., Jensen H., Oliveira L. & Sibani P. 2004. Evolution in complex systems. *Complexity* 10: 49-56.

Argos 2008. *Argos User's Manual*. Toulouse, France: CLS/Service Argos.

Arnould J.P.Y. & Hindell M.A. 2001. Dive behaviour, foraging locations, and maternal-attendance patterns of Australian fur seals (Arctocephalus pusillus doriferus). *Canadian Journal of Zoology* 79: 35-48.

Austin D., Bowen W. & Mcmillan J. 2004. Intraspecific variation in movement patterns: modeling individual behaviour in a large marine predator. *Oikos* 105: 15-30.

Austin D., Mcmillan J.I. & Bowen W.D. 2003. A three-stage algorithm for filtering erroneous Argos satellite locations. *Marine Mammal Science* 19: 371-383.

Bak P., Tang C. & Wiesenfeld K. 1988. Self-organized criticality. *Physical Review A* 38: 364-374.

BBN Technologies 2003. OpenMap.

Beazley D.M. 2006. *Python Essential Reference (3rd Edition)*. Sams.

Blanchette J. & Summerfield M. 2008. C++ GUI programming with Qt 4.

Blum C. & Roli A. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)* 35: 268-308.

Blundell G., Maier J. & Debevec E. 2001. Linear home ranges: Effects of smoothing, sample size, and autocorrelation on kernel estimates. *Ecological Monographs* 71: 469-489.

Boehlert G.W., Costa D.P., Crocker D.E., Green P., O'Brien T., Levitus S. & Le Boeuf B.J. 2001. Autonomous pinniped environmental samplers: Using instrumented animals as oceanographic data collectors. *Journal of Atmospheric and Oceanic* 18: 1882-1893.

Boettcher S. & Percus A. 2000. Nature's way of optimizing. *Artificial Intelligence* 119: 275-286.

Boettcher S. & Percus A.G. 2004. Extremal optimization at the phase transition of the three-coloring problem. *Physical Review E* 69: 66703.

Boettcher S. & Percus A.G. 2001. Optimization with Extremal Dynamics. *Physical Review Letters* 86: 5211–5214.

Bonadonna F., Lea M. & Guinet C. 2000. Foraging routes of Antarctic fur seals (Arctocephalus gazella) investigated by the concurrent use of satellite tracking and time-depth recorders. *Polar Biology* 23: 149-159.

Boveng P.L., Walker B.G. & Bengtson J. 1996. Variability in Antarctic fur seal dive data: Implications for TDR studies. *Marine Mammal Science* 12: 543-554.

Boyd I.L., Reid K. & Bevan R.M. 1995. Swimming speed and allocation of time during the dive cycle in Antarctic fur seals. *Animal Behaviour* 50: 769 - 784.

Bradshaw C., Sims D. & Hays G. 2007. Measurement error causes scale-dependent threshold erosion of biological signals in animal movement data. *Ecological Applications* 17: 628-638.

Burns J.M., Costa D.P., Fedak M.A., Hindell M.A., Bradshaw C.J.A., Gales N.J., Mcdonald B., Trumble S.J. & Crocker D.E. 2004. Winter habitat use and foraging behavior of crabeater seals along the Western Antarctic Peninsula. *Deep-Sea Research Part II: Topical Studies In Oceanography* 51: 2279-2303.

Burns J. & Castellini M. 1998. Dive data from satellite tags and time-depth recorders: A comparison in weddell seal pups. *Marine Mammal Science* 14: 750-764.

Burt W. 1943. Territoriality and home range concepts as applied to mammals. *Journal of Mammalogy* 24: 346-352.

Calenge C. 2007. Exploring habitat selection by wildlife with adehabitat. *Journal of Statistical Software* 22: 2–19.

Callahan S., Freire J., Santos E., Scheidegger C., Silva C. & Vo H. 2006. VisTrails: visualization meets data management. *Proceedings of the 2006 ACM SIGMOD international conference on Management of data* 745-747.

Campagna C., Werner R., Karesh W., Marin M.R., Koontz F., Cook R. & Koontz C. 2001. Movements and location at sea of South American sea lions (Otaria flavescens). *Journal of Zoology* 255: 205-220.

Cardona L., Revelles M., Carreras C., San Felix M., Gazo M. & Aguilar A. 2005. Western Mediterranean immature loggerhead turtles: habitat use in spring and summer assessed through satellite tracking and aerial surveys. *Marine Biology* 147: 583-591.

Cauzac J.P. & Ortega C. 2000. Protecting and Preserving the Earth with ARGOS. *Proceedings of the International Symposium 'GEOMARK 2000'*

Chen M.R., Lu Y.Z. & Yang G. 2007. Population-based extremal optimization with adaptive Lévy mutation for constrained optimization. *Computational Intelligence and Security* 144-155.

Chiong R. 2009. *Nature-inspired algorithms for optimisation*.

Costa D.P., Robinson P., Arnould J., Harrison A., Simmons S., Hassrick J., Hoskins A., Kirkman S., Oosthuizen H., Villegas-Amtmann S. & Crocker D. 2010. Accuracy of ARGOS locations of Pinnipeds at-sea estimated using Fastloc GPS. *PLoS One* 5:

Coyne M.S. & Godley B.J. 2005. Satellite Tracking and Analysis Tool(STAT): an integrated system for archiving, analyzing and mapping animal tracking data. *Marine Ecology Progress Series* 301: 1-7.

Cunningham L., Baxter J.M., Boyd I., Duck C., Lonergan M., Moss S. & Mcconnell B. 2009. Harbour seal movements and haul-out patterns: implications for monitoring and management. *Aquatic Conservation: Marine and Freshwater Ecosystems* 19: 398-407.

Darwin I.F. 2001. *Java Cookbook*. Sebastopol, CA, USA: O'Reilly & Associates, Inc.

Deagle B., Gales N. & Hindell M. 2008. Variability in foraging behaviour of chick-rearing macaroni penguins Eudyptes chrysolophus and its relation to diet. *Marine Ecology Progress Series* 359: 295.

Associates D.& 2003. Mckoi SQL Database.

Eckert S. & Stewart B. 2001. Telemetry and satellite tracking of whale sharks, Rhincodon typus, in the Sea of Cortez, Mexico, and the north Pacific Ocean. *Environmental Biology of Fishes* 60: 299-308.

Edrén S., Wisz M.S., Teilmann J., Dietz R. & Söderkvist J. 2010. Modelling spatial patterns in harbour porpoise satellite telemetry data using maximum entropy. *Ecography* 33: 698-708.

Eric J., Travis O. & Pearu P. 2001. SciPy: Open source scientific tools for Python.

Fauchald P. & Tveraa T. 2003. Using first-passage time in the analysis of area-restricted search and habitat selection. *Ecology* 84: 282-288.

Fedak M.A., Lovell P. & Grant S.M. 2001. Two approaches to compressing and interpreting time-depth information as collected by time-depth recorders and satellite-linked data recorders. *Marine Mammal Science* 17: 94-110.

Fieberg J. 2007. Utilization Distribution Estimation Using Weighted Kernel Density Estimators. *Journal of Wildlife Management* 71: 1669-1675.

Field I.C., Bradshaw C.J.A., Burton H.R. & Hindell M.A. 2004. Seasonal use of oceanographic and fisheries management zones by juvenile southern elephant seals (Mirounga leonina) from Macquarie Island. *Polar Biology* 27: 432-440.

Fieldsend J. & Singh S. 2002. A multi-objective algorithm based upon particle swarm optimisation. *U.K. Workshop on Computational Intelligence* 34-44.

Flanagan D. 2005. *Java In A Nutshell, 5th Edition*. O'Reilly Media, Inc.

Fonseca C.M. & Fleming P.J. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. *Proceedings of the fifth international conference on genetic algorithms* 423: 416-423.

Fowler M. & K. Scott 2000. *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Freitas C., Lydersen C., Fedak M. & Kovacs K. 2008. A simple new algorithm to filter marine mammal Argos locations. *Marine Mammal Science* 24: 315-325.

Frydman S. & Gales N. 2007. HeardMap: Tracking marine vertebrate populations in near real time. *Deep Sea Research Part II: Topical Studies in Oceanography* 54: 384-391.

Galassi M., Davies J., Theiler J., Gough B., Jungman G., Alken P., Booth M., Rossi F. & Price R. 2009. GNU scientific library reference manual: (v. 1.12). *Network Theory Ltd*

Gamma E., R. Helm, R. Johnson & J. Vlissides 1995. *Design patterns: elements of reusable object-oriented software*. Addison-wesley Reading, MA.

Georges J.Y., Bonadonna F. & Guinet C. 2000. Foraging habitat and diving activity of lactating Subantarctic fur seals in relation to sea-surface temperatures at Amsterdam Island. *Marine Ecology Progress Series* 196: 291-304.

Goulet A.M., Hammill M.O. & Barrette C. 1999. Quality of satellite telemetry locations of gray seals (Halichoerus grypus). *Marine Mammal Science* 15: 589-594.

Guinet C., Dubroca L., Lea M.A., Goldsworthy S., Cherel Y., Duhamel G., Bonadonna F. & Donnay J.P. 2001. Spatial distribution of foraging in female Antarctic fur seals Arctocephalus gazella in relation to oceanographic variables: a scale-dependent approach using geographic information systems. *Marine Ecology Progress Series* 219: 251-264.

Hart K., Zawada D., Fujisaki I. & Lidz B. 2010. Inter-nesting habitat-use patterns of loggerhead sea turtles: enhancing satellite tracking with benthic mapping. *Aquatic Biology* 11: 77-90.

Hays G.C. 2004. First records of oceanic dive profiles for leatherback turtles, *Dermochelys coriacea*, indicate behavioural plasticity associated with long-distance migration. *Animal Behaviour* 67: 733-743.

Hays G.C., Akesson S., Godley B.J., Luschi P. & Santidrian P. 2001. The implications of location accuracy for the interpretation of satellite-tracking data. *Animal Behaviour* 61: 1035-1040.

Hedd A., Gales R. & Brothers N. 2001. Foraging strategies of shy albatross Thalassarche cauta breeding at Albatross Island, Tasmania, Australia. *Marine Ecology Progress Series* 224: 267-282.

Henderson A., J. Ahrens & C. Law 2004. *The Paraview Guide*. Kitware.

Hindell M. & Lea M. 1998. Heart rate, swimming speed, and estimated oxygen consumption of a free-ranging southern elephant seal. *Physiological Zoology* 71: 74-84.

Hines W.G.S., O'Hara H., R. J., Pond B. & Obbard M.E. 2005. Allowing for redundancy and environmental effects in estimates of home range utilization distributions. *Environmetrics* 16: 33-50.

Horne J.S. & Garton E. 2006a. Likelihood cross-validation versus least squares cross-validation for choosing the smoothing parameter in kernel home-range analysis. *Journal of Wildlife Management* 70: 641-648.

Horne J. & Garton E. 2006b. Selecting the best home range model: an information-theoretic approach. *Ecology* 87: 1146-1152.

Horne J., Garton E., Krone S. & Lewis J. 2007. Analyzing animal movements using Brownian bridges. *Ecology* 88: 2354-2363.

Hull C.L., Hindell M.A. & Michael K. 1997. Foraging zones of royal penguins during the breeding season, and their association with oceanographic features. *Marine Ecology Progress Series* 153: 217-228.

Hunter J.D. 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science and Engineering* 9: 90–95.

James M.C., Andrea O., C. & Myers R.A. 2005. Identification of high-use habitat and threats to leatherback sea turtles in northern waters: new directions for conservation. *Ecology Letters* 8: 195-201.

Jonker F.C. & Bester M.N. 1998. Seasonal movements and foraging areas of adult southern female elephant seals, Mirounga leonina, from Marion Island. *Antarctic Science* 10: 21-30.

Jonsen I.D., Flenming J. & Myers R. 2005. Robust state-space modeling of animal movement data. *Ecology* 86: 2874-2880.

Jonsen I.D., Myers R.A. & Flemming J. 2003. Meta-analysis of animal movement using state-space models. *Ecology* 84: 3055-3063.

Juneau J., J. Baker, L. Soto, F. Wierzbicki & V. Ng 2010. *The definitive guide to Jython: Python for the Java platform*. Apress.

Karaaslan U., Varaiya P. & Walrand J. 2009. Two proposals to improve freeway traffic flow. *American Control Conference, 1991* 2539-2544.

Katajisto J. & Moilanen A. 2006. Kernel-based home range method for data with irregular sampling intervals. *Ecological Modelling* 194: 405-413.

Keating K. & Cherry S. 2009. Modeling utilization distributions in space and time. *Ecology* 90: 1971-1980.

Kernighan B., D. Ritchie & P. Ejeklint 1978. *The C programming language*. Citeseer.

Kirkpatrick S., Gelatt C.D., Jr. & Vecchi M.P. 1983. Optimization by Simulated Annealing. *Science* 220: 671-680.

Kissock J.K. & J. Seryak 2004. *Understanding Manufacturing Energy Use Through Statistical Analysis*. Texas A&M University.

Kuhn C., Mcdonald B., Shaffer S.A., Barnes J., Crocker D.E., Burns J. & Costa D. 2006. Diving physiology and winter foraging behavior of a juvenile leopard seal (Hydrurga leptonyx). *Polar Biology* 29: 303-307.

Lawrence M.A. 2011. Easy analysis and visualization of factorial experiments. *http://groups.google.com/group/ez4r*

Le Boeuf B.J., Crocker D.E., Costa D.P., Blackwell S.B., Webb P.M. & Houser D.S. 2000. Foraging ecology of northern elephant seals. *Ecological Monographs* 70: 353-382.

Lesage W., Harnmill M.O. & Kovacs K.M. 2004. Long-distance movements of harbour seals (Phoca vitulina) from a seasonally ice-covered area, the St. Lawrence River estuary, Canada. *Canadian Journal of Zoology* 82: 1070-1081.

Lonergan M., Fedak M. & Mcconnell B. 2009. The effects of interpolation error and location quality on animal track reconstruction. *Marine Mammal Science* 25: 275-282.

Lunardi A. 2009. *Interpolation theory*. Pisa: Appunti.

Marco D. & D.C. Gianni 1999. *The ant colony optimization meta-heuristic*. Maidenhead, UK, England: McGraw-Hill Ltd., UK.

Matthiopoulos J., Mcconnell B., Duck C. & Fedak M. 2004. Using satellite telemetry and aerial counts to estimate space use by grey seals around the British Isles. *Journal of Animal Ecology* 41: 476-491.

Matthiopoulos J. 2003. Model-supervised kernel smoothing for the estimation of spatial usage. *Oikos* 102: 367-377.

Mcconnell B.J., Chambers C. & Fedak M.A. 1992. Foraging ecology of Southern Elephant seals in relation to the bathymetry and productivity of the Southern ocean. *Antarctic Science* 4: 393-398.

Mcneil D. 1992. On graphing paired data. *American Statistician* 46: 307-311.

Mitchell M. 1998. *An Introduction to Genetic Algorithms*. The MIT Press.

Moler C.B. 2004. *Numerical Computing with Matlab*. Society for Industrial Mathematics.

Moore J. & Chapman R. 1999. Application of Particle Swarm to Multiobjective Optimization.

Naito Y., Asaga T. & Ohyama Y. 1990. Diving Behavior of Adelie Penguins Determined by Time-Depth Recorder. *Condor* 92: 582-586.

Oliphant T.E. 2006. *Guide to NumPy*. Provo, UT: Brigham Young University.

Osyczka A. 2002. *Evolutionary algorithms for single and multicriteria design optimization*. New York: Physica-Verlag.

Owens M. & M. Owens 2006. *The definitive guide to SQLite*. Apress.

Paczuski M., Maslov S. & Bak P. 1996. Avalanche dynamics in evolution, growth, and depinning models. *Physical Review E* 53: 414-443.

Patchett J., Ahrens J. & Ahern… S. 2009. Parallel Visualization and Analysis with ParaView on a Cray XT4. *www.osti.gov*

Patterson T., Mcconnell B., Fedak M., Bravington M. & Hindell M. 2010. Using GPS data to evaluate the accuracy of state-space methods for correction of Argos satellite telemetry error. *Ecology* 91: 273-285.

Patterson T., Thomas L., Wilcox C., Ovaskainen O. & Matthiopoulos J. 2008. State-space models of individual animal movement. *Trends in Ecology and Evolution* 23: 87-94.

Paul F.D. 2007. Guest Editor's Introduction: Python: Batteries Included. *Computing in Science and Engineering* 9: 7-9.

Phillips R.A., Silk J.R.D., Croxall J.P., Afanasyev V. & Briggs D.R. 2004. Accuracy of geolocation estimates for flying seabirds. *Marine Ecology Progress Series* 266: 265-272.

Pilgrim M. 2010. *Dive Into Python 3*. Paramount, CA: CreateSpace.

Pinaud D. & Weimerskirch H. 2005. Scale-dependent habitat use in a long-ranging central place predator. *Journal of Animal Ecology* 74: 852-863.

R D.C.T. 2008. R: A Language and Environment for Statistical Computing.

Ripley B.D. 2001. The R project in statistical computing. *MSOR Connections. The newsletter of the LTSN Maths, Stats & OR Network* 1: 23-25.

Robinson S., Mckinlay J., Gales N., Candy S., Casper R., Goldsworthy S., Staniland I. & Frydman S. In Prep. Foraging behaviour of lactating Antarctic fur seals at Heard Island: determining daily meso-scale dive behaviour.

Robinson S.A., Goldsworthy S.G., Van Den Hoff J. & Hindell M.A. 2002. The foraging ecology of two sympatric fur seal species, Arctocephalus gazella and Arctocephalus tropicalis, at Macquarie Island during the austral summer. *Marine and Freshwater Research* 53: 1071-1082.

Rooney S.M., Wolfe A. & Hayden T.J. 1998. Autocorrelated data in telemetry studies: time to independence and the problem of behavioural effects. *Mammal Review* 28: 89-98.

Royer F. & Lutcavage M. 2008. Filtering and interpreting location errors in satellite telemetry of marine animals. *Journal of Experimental Marine Biology and Ecology* 359: 1-10.

Rutz C. & Hays G.C. 2009. New frontiers in biologging science. *Biology Letters* 5: 289-292.

Schroeder W., K. Martin, B. Lorensen, L. Avila, R. Avila & C.C. Law 1998. *The Visualization Toolkit*. Prentice Hall PTR.

Seaman D., Griffith B. & Powell R. 1998. KERNELHR: a program for estimating animal home ranges. *Wildlife Society Bulletin* 26: 95-100.

Seaman D., Millspaugh J., Kernohan B., Brundige G., Raedeke K. & Gitzen R. 1999. Effects of sample size on kernel home range estimates. *The Journal of Wildlife Management* 739-747.

Seaman D. & Powell R.A. 1996. An evaluation of the accuracy of kernel density estimators for home range analysis. *Ecology* 77: 2075-2085.

Sean L. 2009. *Essentials of Metaheuristics*.

Seney E. & Landry A. 2008. Movements of Kemp❼s ridley sea turtles nesting on the upper Texas coast: implications for management. *Endangered Species* 4: 73-84.

Silverman B. 1986. Density Estimation for Statistics and Data Analysis. Number 26 in Monographs on statistics and applied probability.

Simonoff J.S. 1996. *Smoothing Methods in Statistics*. New York: Springer-Verlag.

Sjoberg M., Fedak M.A. & Mcconnell B.J. 1995. Movements And Diurnal Behavior Patterns In A Baltic Grey Seal (Halichoerus-Grypus). *Polar Biology* 15: 593-595.

Skern-Mauritzen M., Kirkman S.P., Olsen E., BjvTRge A., Drapeau L., MevØer M.A., Roux J.-P., Swanson S. & Oosthuizen W.H. 2009. Do inter-colony differences in Cape fur seal foraging behaviour reflect large-scale changes in the northern Benguela ecosystem? *African Journal of Marine Science* 31: 399 - 408.

Standish T. 1994. *Data Structures, Algorithms, & Software Principles in C*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Stroustrup B. 1997. The C++ programming language.

Sumner M., Wotherspoon S. & Hindell M. 2009. Bayesian estimation of animal movement from archival and satellite tags. *PLoS One* 4: e7324.

Thacker N.A., Aherne F.J. & Rockett P.I. 1997. The Bhattacharyya metric as an absolute similarity measure for frequency coded data. *Kybernetika* 34: 363-368.

Thompson D., Moss S. & Lovell P. 2003a. Foraging behaviour of South American fur seals Arctocephalus australis: extracting fine scale foraging behaviour from satellite tracks. *Marine Ecology Progress Series* 260: 285-296.

Thompson D., Moss S.E.W. & Lovell P. 2003b. Foraging behaviour of South American fur seals Arctocephalus australis: extracting fine scale foraging behaviour from satellite tracks. *Marine Ecology Progress Series* 260: 285-296.

Tiwari G. 2000. Traffic flow and safety: need for new models for heterogeneous traffic. *Injury prevention and control* 71:

Tremblay Y., Roberts A. & Costa D.P. 2007. Fractal landscape method: an alternative approach to measuring area-restricted searching behavior. *Journal of Experimental Biology* 210: 935-945.

Tremblay Y., Shaffer S., Fowler S., Kuhn C., Mcdonald B., Weise M., Bost C., Weimerskirch H., Crocker D., Goebel M. & Costa D. 2006. Interpolation of animal tracking data in a fluid environment. *Journal of Experimental Biology* 209: 128-140.

Tremblay Y., Robinson P.W. & Costa D.P. 2009. A Parsimonious Approach to Modeling Animal Movement Data. *PLoS One* 4: e4711.

Turchin P. 1998. *Quantitative analysis of movement: measuring and modeling population redistribution in plants and animals*. Sunderland, MA: Sinauer Associates.

Van Der Weken D., Nachtegael M. & Kerre E. 2003. Using similarity measures for histogram comparison. *Fuzzy Sets and Systems - IFSA 2003* 1-9.

Venables W.N. & D.M. Smith 2009. *An Introduction to R*. Network Theory Ltd.

Vincent C., Mcconnell B.J., Ridoux V. & Fedak M.A. 2002. Assessment of Argos location accuracy from satellite tags deployed on captive gray seals. *Marine Mammal Science* 18: 156-166.

Vokoun J. 2003. Kernel density estimates of linear home ranges for stream fishes: Advantages and data requirements. *North American Journal of Fisheries Management* 23: 1020-1029.

Walters E.G. 2001. *The essential guide to computing*. Prentice Hall PTR.

Wienecke B. & Robertson G. 2006. Comparison of foraging strategies of incubating king penguins Aptenodytes patagonicus from Macquarie and Heard islands. *Polar Biology* 29: 424-438.

Wildlife Computers. 2003. Instrument Helper.

Williams E. 2010. Aviation Formulary V1.45.

Wilson R.P., Ropert-Coudert Y. & Kato A. 2002. Rush and grab strategies in foraging marine endotherms: the case for haste in penguins. *Animal Behaviour* 63: 85-95.

Wood A.G., Naef-Daenzer B., Prince P. & Croxall J. 2000. Quantifying habitat use in satellite-tracked pelagic seabirds: application of kernel estimation to albatross locations. *Journal of Avian Biology* 31: 278-286.

Worton B.J. 1995. Using Monte-Carlo simulation to evaluate kernel-based home-range estimators. *Journal of Wildlife Management* 59: 794-800.

Worton B. 1989. Kernel methods for estimating the utilization distribution in home-range studies. *Ecology* 70: 164-168.

Zar J.H. 2009. *Biostatistical Analysis (5th Edition)*. Prentice Hall.

Zwillinger D. 2003. *CRC standard mathematical tables and formulae*. CRC press.